A Quadratic-Time Response Time Upper Bound with a Tightness Property

Enrico Bini*, Andrea Parri*, Giacomo Dossena
*Scuola Superiore Sant'Anna, Pisa, Italy

***Abstract*—The response time analysis (RTA) is one of the fundamental tools used to guarantee the schedulability of sets of real-time tasks scheduled by Fixed Priorities. Also, several analysis methods inspired by RTA have been successfully developed to address more sophisticated execution platforms (distributed systems, multiprocessor) and application models (DAGs). The major issue with RTA is its time complexity, which is NP-hard. Such a complexity shows up when the task set has high utilization and RTA needs to check all jobs until the first idle instant.**

In this paper, we propose a continuous upper bound to the response time with quadratic time complexity in the number of tasks. Such an upper bound is demonstrated to be tighter than previously proposed ones with linear time complexity. In addition, with two tasks only, we prove that the proposed bound is the tightest continuous function upper bounding the exact response time of sets of tasks with full utilization. Whether or not this property holds with more than two tasks is still an open problem.

1. Introduction

Real-time systems arise in all applications in which the timing of computation is as important as the result itself. Examples of application domains are automotive, avionics, multimedia, process control, etc. The distinguishing feature of real-time systems is the necessity to determine the completion time of tasks. Despite the advancement of computing architectures from single core to multi core, heterogeneous platforms, or distributed systems, a fundamental methodology for the analysis of real-time systems is the Response Time Analysis (RTA) [1], [2], [3].

Although being developed for single cores only, RTA plays a key role in the post single-core era as well. RTA can be used in a straightforward manner on multi-core scheduling problems, when tasks are partitioned among the cores, by formulating the partitioning as an Integer Linear Programming (ILP) problem [4]. In this case, a multi-core scheduling problem is reduced to m single-core scheduling problems, with m equal to the number of cores. Similar arguments are also used in global multi-core scheduling algorithms, where tasks are allowed to migrate among cores. RTA-like methods [5], [6] are used to compute the maximum interference on the execution of each task and then the response time.

In distributed systems, RTA is used to estimate the task completion time at each node [7], [8]. Then the task completion times determine the activation jitter of the tasks (or messages over a communication medium) whose activation depends on the completion of preceding tasks.

RTA, however, has the following weaknesses.

1) Complexity: RTA is demonstrated to be NP-hard [9]. The complexity of RTA makes it hardly applicable to cases in which tasks may arrive and leave during run-time. The time complexity of RTA is an even greater hurdle in all the analysis methods which require the iteration of RTA, such as the holistic analysis in distributed systems [7], [8].
   The running time of RTA grows when the system is highly loaded. At high loads, in fact, the longest response time may not occur at the first job and the entire busy period must be checked [10]. An illustration of this fact is given in Figure 1. The figure reports the average number of iterations of RTA as a function of the total system load, for 100.000 random task sets. The three plots correspond to three different values of task periods $\{T_1, T_2, T_3\}$:
   - $\{8, 12, 24\}$ in solid black,
   - $\{8, 15, 20\}$ in dashed black, and
   - $\{7, 15, 26\}$ in gray.

   While at low utilization value the average number of iterations of the three period configurations are comparable, at utilizations closer to 1 the RTA run-time grows with the least common multiple of the periods (called *hyperperiod*), unless some additional hypothesis, such as *constrained deadline*, holds. In fact, when the hyperperiod is long, the number of jobs within the busy period grows. The picture does not qualitatively change for other settings.
   We remark that high-load task sets (for which RTA run-time is long) are those enabling a more efficient usage of the available computing capacity. Hence, they are relevant in all resource-constrained applications.
2) Presence of discontinuities: The task response time is a discontinuous function of the task parameters. Hence, an arbitrary small perturbation on any input of the problem, which is the representation of the task set, may produce a tangible variation of the response time. Hence, we believe that the analysis and the design of critical applications, as real-time systems are, should rely on robust analysis methods

since variations on parameters may indeed happen, for example due to clock/timer drift.

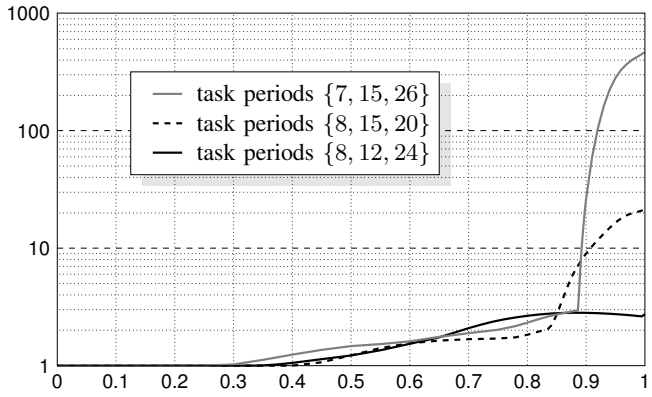

Figure 1. Average number of RTA iterations as a function of system load.

We highlight that both points above ("Complexity" and "Presence of discontinuities") represent intrinsic characteristics of the response time computation itself, i.e., these properties do *not* depend on the particular method that is used to compute the response time.

Below, we provide a comparison against the vast body of related literature.

1.1. Related works and our contribution

RTA is widely used in the real-time research community. A considerable amount of work has been developed to address the above mentioned issues.

A notable result on the time complexity of RTA is due to Eisenbrand and Rothvoß, who showed that RTA is NP-hard [9]. However, if the additional hypothesis of harmonic periods is given, meaning that the periods of the tasks pairwise divide each other, then a polynomial-time algorithm exists [11].

Taking advantage of an approximation of the interference [12], [13], a Fully Polynomial-Time Approximation Scheme (FPTAS) for computing response time upper-bounds under resource augmentation, in presence of arbitrary deadlines, was proposed by Nguyen et al. [14], [15]. In their work the following resource augmentation guarantee is provided: if the test with accuracy $\epsilon \in (0, 1)$ responds that the task set is not schedulable, then the task set is certainly not schedulable over a processor of speed $1-\epsilon$. The above listed methods, however, compute the task response time as a function of the task set parameters, which is still discontinuous.

Several authors independently proposed a linear-time continuous response time upper bound [16], [17], [18]. Also, this bound was used [19] to compute a utilization upper bound for arbitraty-deadline tasks, tighter than Lehoczky's one [10]. The linear-time response time bound, however, may be arbitrarily distant to the actual response time, especially when task periods are harmonic [18]. Okwudire et al. [20] then improved the upper bound by replacing tasks with harmonic periods with an artificial task, and then providing a tighter upper bound to the interference by higher priority tasks. In a recent report, Chen, Huang, Liu [21] proposed a general framework for bounding the interference of higher priority tasks, which can be used to derive an upper bound to the response time (special case of our Theorem 1).

Another response time upper bound, with complexity $O(n \log n)$ in the number n of tasks, is due to Baruah [22]. This bound, however, is limited to the case of tasks with total utilization less than the Liu and Layland bound [23] of $\ln 2$, while ours does not have such a restriction. Also notice that, when the total utilization is less than the Liu and Layland utilization bound, the RTA run time is a less stringent issue as only the first task release has to be checked.

Contribution of the paper. To overcome the limitations discussed so far, we propose an upper bound to the response time with the following properties.

- *Quadratic time complexity* in the number of tasks. Such a complexity is independent of the task set load, making the response time upper bound particularly useful for a task set with high load.
- *Continuity* and *differentiability* in all the task set parameters. This property can drastically speed-up the ILP-based solvers of optimal real-time design problems. In classic problem formulations [24], [25], the response time is expressed as a linear combination of integer variables representing the number of interfering jobs. By using our proposed bound, these integer variables are not needed and solvers may run faster.
- *Tightness at full utilization*. The proposed bound is the lowest possible continuous function upper bounding the response time of full-utilization sets of tasks. We can make a small perturbation to the parameters such that the exact response time of the perturbed task set is as close as desired to the upper bound of the original task set. This property is proved for sets of two tasks. We were unable to extend this result to an arbitrary number of tasks, but we conducted numerical simulations suggesting that this may be the case.
- The bound is extended to the case with task *blocking time*, *jitter*, and *cache related preemption delay*.

The paper is organized as follows. In Section 2, we introduce our system model and notation, and we recall some basics of RTA. In Section 3 we derive the response time upper bound. Section 4 investigates the tightness of the bound: it is proved that the bound is the lowest continuous function upper bounding the response time of a full utilization set of two tasks. In Section 5 we extend the bound to the case with jitter, blocking, and cache-related preemption delay, and finally in Section 6 we conclude the paper describing some open problems.

2. System model and background

We model a real-time application by a set of n sporadic tasks. The i-th task is modeled by

- a *computation time* $C_i > 0$,
- a *minimum interarrival time* (also called *period*, for brevity) $T_i > 0$, and
- a *deadline* D_i, which may be unrelated to the task period (arbitrary deadline model).

We denote by $U_i := \frac{C_i}{T_i}$ the *utilization* of the i-th task. Intuitively, U_i represents the fraction of time consumed by the i-th task.

At time instants denoted by $a_{i,j}$, the i-th task requests the execution of its j-th *job* for an amount C_i of time. Two consecutive requests for job execution cannot be separated by less than T_i, that is

$$\forall i, j, \qquad a_{i,j+1} \geq a_{i,j} + T_i.$$

We denote the *finishing time* of the j-th job of the i-th task by $f_{i,j}$. Thanks to these basic notions, we can define both the job and the task response time.

Definition 1 (Job response time). Let $a_{i,j}$ and $f_{i,j}$ be respectively the activation and the finishing times of the j-th job of the i-th task. Then, the *job response time* $R_{i,j}$ of the j-th job of the i-th task is defined as

$$R_{i,j} := f_{i,j} - a_{i,j}. \tag{1}$$

Definition 2 (Task response time). Let $R_{i,j}$ be the response time of the j-th job of the i-th task. Then, the *response time* R_i of the i-th task is defined as

$$R_i := \max_{j=1,2,\ldots} R_{i,j}. \tag{2}$$

A task set is said to be *schedulable* when

$$\forall i, \qquad R_i \leq D_i.$$

In this paper, we consider the case of tasks scheduled by a *preemptive fixed-priority* scheduler over a *single processor*: any job of a task executes only if all jobs released by higher priority tasks have completed. Tasks are sorted by *decreasing* priority: $i < j$ if and only if the i-th task has priority higher than the j-th task; jobs released by the same task are served in First-Come-First-Served order.

Without loss of generality, from now on we investigate the *response time* R_n of the lowest priority task. In fact, due to the preemptive and fixed-priority nature of the scheduler, the response time R_i of any other task i with $i < n$ is equal to the response time of the lowest priority task of a set in which all tasks from the $(i+1)$-th to the n-th are removed and n is set equal to i.

In single-processor periodic-task scheduling, as proved by Liu and Layland [23], the longest response time occurs when all higher priority tasks are simultaneously activated (quoting Theorem 1 in [23]: "A critical instant for any task occurs whenever the task is requested simultaneously with requests for all higher priority tasks."). Without loss of generality, we call "0" such a *critical instant* of simultaneous activations of all tasks and we start numbering the task jobs from 1 at time 0, that is

$$\forall i, \qquad a_{i,1} = 0. \tag{3}$$

In fixed-priority scheduling over a single processor, the activation pattern of a sporadic task that maximizes the interference over the n-th task always consists of jobs being activated as soon as possible. Then, for the purpose of computing the longest response time of (2), we assume that the job activations are

$$a_{i,j} = (j-1)T_i. \tag{4}$$

With these hypothesis and notation, a standard way [1], [2] to compute the response time R_n is through the following recurrence relation

$$\begin{cases} t_0 = 0 \\ t_{k+1} = g(c, t_k) \end{cases} \tag{5}$$

with

$$g(c,t) = c + \sum_{i=1}^{n-1} \left\lceil \frac{t}{T_i} \right\rceil C_i. \tag{6}$$

The recurrence of (5) is proved to terminate [1] in a finite number of steps $\forall c \geq 0$ if $\sum_{i=1}^{n-1} U_i < 1$. Hence, we denote by $t_*(c)$ the convergence point of the recurrence (5), for any value of $c \geq 0$. With this notation, the response time R_n of (2) is equal to [10], [26]

$$R_n = \max_{j \in \mathcal{J}^*} \{ \underbrace{t_*(jC_n)}_{f_{n,j}} - \underbrace{(j-1)T_n}_{a_{n,j}} \}. \tag{7}$$

with $\mathcal{J}^*$ being the set of job indices of the n-th task belonging to the *level-n busy period* [10], that is

$$\mathcal{J}^* = \{ j \in \mathbb{N} \ : \ j \geq 1, \ t_*((j-1)C_n) \geq (j-1)T_n \}. \tag{8}$$

Notice that $\mathcal{J}^*$ is never empty, as it is always $1 \in \mathcal{J}^*$.

In words, for all $j \in \mathcal{J}^*$ the $(j-1)$-th job of the n-th task finishes (at $f_{i,j-1} = t_*((j-1)C_n)$) later than or at the activation of the j-th job of the n-th task (at $a_{i,j} = (j-1)T_n$). This means that the scheduler never idles in $[0, t_*(jC_n)]$ for all $j \in \mathcal{J}^*$. As shown by Lehoczky [10], the longest response time originating the maximum of (2) may occur at any job in $\mathcal{J}^*$. Hence, all jobs $j \in \mathcal{J}^*$ must be checked, as required in (7).

3. Response time upper bound

The difficulty in computing the response time of a task lies in solving the fixed-point equation in (5). Additionally, whenever the level-n busy period contains more than one job of the n-th task (that is $|\mathcal{J}^*| > 1$), the computation of the fixed point $t_*(jC_n)$ required in (7) must be performed for all $j \in \mathcal{J}^*$. In this section, we illustrate a quadratic-time method to compute an upper bound to the response time (rather than its exact value).

Since $t_*(jC_n)$ is the fixed-point of convergence of (5), it can also be written as

$$t_*(jC_n) = jC_n + \sum_{i=1}^{n-1} \left\lceil \frac{t_*(jC_n)}{T_i} \right\rceil C_i. \tag{9}$$

Let us now perform the Euclidean division of $t_*(jC_n)$ by the period T_i of any higher priority task, by defining the quotient $q_i^{(j)}$ and the remainder $r_i^{(j)}$ as follows

$$\begin{aligned} q_i^{(j)} &:= \left\lceil \frac{t_*(jC_n)}{T_i} \right\rceil \geq 1 \\ r_i^{(j)} &:= \left\lceil \frac{t_*(jC_n)}{T_i} \right\rceil T_i - t_*(jC_n) = q_i^{(j)} T_i - t_*(jC_n), \end{aligned} \tag{10}$$

for every high priority task $i = 1, \ldots, n-1$ and job $j \in \mathcal{J}^*$ of the n-th task. With these definitions, the fixed point $t_*(jC_n)$ can be written as a multiple of the period T_i of any task minus the remainder as

$$\forall i = 1, \ldots, n-1, \qquad t_*(jC_n) = q_i^{(j)} T_i - r_i^{(j)}, \tag{11}$$

with the usual property of a remainder, $r_i^{(j)} \in [0, T_i)$. Notice that, differently than the classic Euclidean division, the remainder in (11) needs to be subtracted (rather than added) because in (10) the quotient is computed by the ceiling (and not the floor) operator.

From the definition of $q_i^{(j)}$ of (10), the fixed point $t_*(jC_n)$ of (9) can also be written as

$$t_*(jC_n) = jC_n + \sum_{i=1}^{n-1} q_i^{(j)} C_i. \tag{12}$$

By multiplying (11) by U_i, we find

$$q_i^{(j)} C_i = t_*(jC_n)\, U_i + r_i^{(j)} U_i,$$

which enables us to rewrite (12) as

$$t_*(jC_n) = jC_n + \sum_{i=1}^{n-1} (t_*(jC_n)\, U_i + r_i^{(j)} U_i),$$

or, equivalently, as

$$t_*(jC_n) = \frac{jC_n + \sum_{i=1}^{n-1} r_i^{(j)} U_i}{1 - \sum_{i=1}^{n-1} U_i}. \tag{13}$$

Notice that although Eq. (13) seems to be a closed-form expression for $t_*(jC_n)$, it is not. Eq. (13) is still a fixed-point equation, as $r_i^{(j)}$ depends on $t_*(jC_n)$ through its definition of (10).

Our strategy to establish an upper bound for $t_*(jC_n)$, and then for the response time R_n of the n-th task through (7), is to find an upper bound to the term $\sum_{i=1}^{n-1} r_i^{(j)} U_i$ in Eq. (13).

Clearly, different upper bounds to $\sum_{i=1}^{n-1} r_i^{(j)} U_i$ yield different upper bounds to R_n. Below, we show the generality of (13), by describing how well-known upper bounds to the response time can be obtained by bounding $\sum_{i=1}^{n-1} r_i^{(j)} U_i$. Later in Section 3.1, we describe a tighter upper bound to the response time, which is the original contribution of this paper.

A simple way to establish an upper bound for $\sum_{i=1}^{n-1} r_i^{(j)} U_i$ is to bound each remainder $r_i^{(j)}$ by T_i, since by the definition (10) of $r_i^{(j)}$, it is always $r_i^{(j)} < T_i$. This leads to a simple upper bound to the fixed point $t_*(jC_n)$, which is also the finishing time of the j-th job of the n-th task for all $j \in \mathcal{J}^*$, that is

$$t_*(jC_n) \leq f'_{n,j} := \frac{jC_n + \sum_{i=1}^{n-1} C_i}{1 - \sum_{i=1}^{n-1} U_i}. \tag{14}$$

The bound to the fixed point $t_*(jC_n)$ of (14) can be used to find an upper bound to the response time R_n of the n-th task, through its definition of (7), as follows

$$R_n \leq \max_{j=1,2,\ldots} \left\{ f'_{n,j} - (j-1)T_n \right\} = \frac{C_n + \sum_{i=1}^{n-1} C_i}{1 - \sum_{i=1}^{n-1} U_i}, \tag{15}$$

since the maximum in (15) is demonstrated [16], [17], [18] to happen at $j = 1$ if and only if $\sum_{i=1}^{n} U_i \leq 1$. Notice that the bound of (15) can be equivalently found by applying the upper bound to $\lceil x \rceil$ with $x + 1$ in (9), as suggested by Sjödin and Hansson [27].

A second tighter bound to the response time can be found by observing that the instant $t_*(jC_n)$, which is the completion time of some job of the n-th task, can never fall in an interval in which some higher priority jobs will be certainly running.

As illustrated in Figure 2, some job with priority higher than the one of the n-th task is always executing in the interval $(q_i^{(j)} T_i, q_i^{(j)} T_i + C_i]$ for any index $i = 1, \ldots, n-1$ of a high priority task: either it is executing the q_i-th job of the i-th task, or some job with priority higher than it. Hence, for any $i = 1, \ldots, n-1$, $t_*(jC_n)$ can be more tightly confined in

$$\left(q_i^{(j)} - 1\right) T_i + C_i < t_*(jC_n) \leq q_i^{(j)} T_i.$$

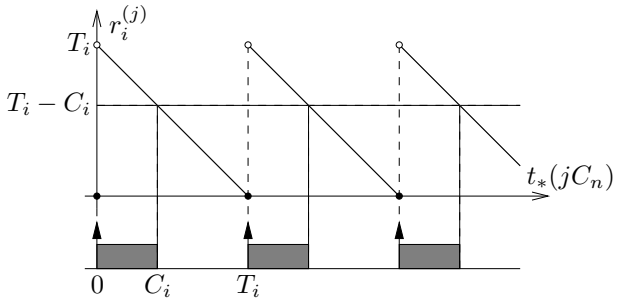

Figure 2. Tightening the upper bound to $r_i^{(j)}$.

Following this intuition, also depicted in Figure 2, from the definition of $r_i^{(j)}$ the following bound on $r_i^{(j)}$ can be

found

$$r_i^{(j)} = q_i^{(j)} T_i - t_*(jC_n) < q_i^{(j)} T_i - \left(q_i^{(j)} - 1\right) T_i - C_i = T_i - C_i, \quad (16)$$

from which we find

$$t_*(jC_n) \le f''_{n,j} := \frac{jC_n + \sum_{i=1}^{n-1}(T_i - C_i)U_i}{1 - \sum_{i=1}^{n-1} U_i} \quad (17)$$

and, for the very same arguments of the proof of (15) demonstrated in [16], [17], [18], the following upper bound to the response time

$$R_n \le \frac{C_n + \sum_{i=1}^{n-1}(T_i - C_i)U_i}{1 - \sum_{i=1}^{n-1} U_i} \quad (18)$$

also holds. Notice that it is (strictly) tighter than the upper bound in (15).

As we explain in the next section, by exploiting even further the reasoning illustrated in Figure 2 it is possible to find a tighter upper bound to the response time.

3.1. Tightening the bound

The bounds shown previously were computed by bounding each $r_i^{(j)}$ individually and then deriving a bound to $\sum_{i=1}^{n-1} r_i^{(j)} U_i$ combining these individual bounds. However, all $r_1^{(j)}, r_2^{(j)}, \ldots r_{n-1}^{(j)}$ are related to each other as they are all function of the same $t_*(jC_n)$. Hence, bounding each $r_i^{(j)}$ independently can indeed introduce some pessimism, since the condition on $t_*(jC_n)$ maximizing some $r_i^{(j)}$ may be incompatible with the condition maximizing $r_\ell^{(j)}$ of some other task. In this section we aim at reducing such a pessimism by bounding $\sum_{i=1}^{n-1} r_i^{(j)} U_i$ all together.

Below we report the main result of this paper.

Theorem 1. Let us define

$$\widehat{R}_n := \frac{C_n + \sum_{i=1}^{n-1}(T_i - C_i)U_i - \beta_n}{1 - \sum_{i=1}^{n-1} U_i}, \quad (19)$$

with

$$\beta_n := \sum_{1 \le j < i \le n-1} \min\{T_i, T_j\} U_i U_j. \quad (20)$$

Then, the response time R_n of the n-th task is bounded from above as follows:

$$R_n \le \widehat{R}_n.$$

Proof: For any job j of the n-th task belonging to the level-n busy period $j \in \mathcal{J}^*$, let $t_*(jC_n)$ be the fixed point of (9), which defines $q_i^{(j)}$ and $r_i^{(j)}$ through (10), for the i-th high priority task, with $i = 1, \ldots, n-1$.

We choose a permutation $\pi : \{1, \ldots, n-1\} \to \{1, \ldots, n-1\}$ ordering the latest releases of all tasks strictly earlier than $t_*(jC_n)$ as follows

$$\left(q_{\pi(n-1)}^{(j)} - 1\right) T_{\pi(n-1)} \le \cdots \le \left(q_{\pi(2)}^{(j)} - 1\right) T_{\pi(2)} \le \left(q_{\pi(1)}^{(j)} - 1\right) T_{\pi(1)} < t_*(jC_n), \quad (21)$$

as also illustrated in Figure 3.

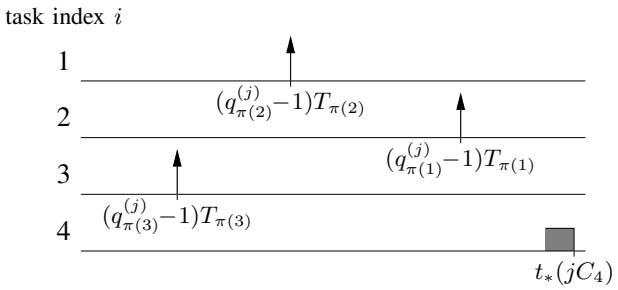

Figure 3. Illustration of the latest activation of high priority tasks earlier than $t_*(jC_n)$ (by an arrow), and the definition of the permutation π. In the figure $n = 4$, $\pi(1) = 2$, $\pi(2) = 1$, and $\pi(3) = 3$.

Now, the crucial observation of the theorem. Since $t_*(jC_n)$ is the finishing time of some job of the n-th task, the n-th task must be scheduled in a left neighborhood of $t_*(jC_n)$ (shown by a gray schedule of the n-th task in Figure 3). Hence, the $n-1$ jobs released at $(q_i^{(j)} - 1)T_i$ must have finished at $t_*(jC_n)$. Since $\left(q_{\pi(1)}^{(j)} - 1\right)T_{\pi(1)}$ is the latest release of high priority jobs earlier than $t_*(jC_n)$, it must necessarily happen that

$$t_*(jC_n) > \left(q_{\pi(1)}^{(j)} - 1\right) T_{\pi(1)} + C_{\pi(1)}.$$

By extending the same reasoning to the other high priority tasks, it must be

$$\begin{aligned} t_*(jC_n) &> \left(q_{\pi(2)}^{(j)} - 1\right) T_{\pi(2)} + C_{\pi(1)} + C_{\pi(2)} \\ &\cdots \quad \cdots \\ t_*(jC_n) &> \left(q_{\pi(n-1)}^{(j)} - 1\right) T_{\pi(n-1)} + \sum_{\ell=1}^{n-1} C_{\pi(\ell)}. \end{aligned}$$

By exploiting the definition of $r_i^{(j)}$ in (10) and the inequalities listed above, an upper bound on $r_i^{(j)}$ follows

$$\begin{aligned} r_{\pi(1)}^{(j)} &< T_{\pi(1)} - C_{\pi(1)} \\ r_{\pi(2)}^{(j)} &< T_{\pi(1)} - \left(C_{\pi(1)} + C_{\pi(2)}\right) \\ &\cdots \quad \cdots \\ r_{\pi(n-1)} &< T_{\pi(n-1)} - \sum_{\ell=1}^{n-1} C_{\pi(\ell)}. \end{aligned}$$

After multiplying each of the above inequalities with the corresponding utilization $U_{\pi(i)}$, and summing over $i = 1, \ldots, n-1$, we obtain

$$\sum_{i=1}^{n-1} r_i^{(j)} U_i = \sum_{i=1}^{n-1} r_{\pi(i)}^{(j)} U_{\pi(i)} \le \sum_{i=1}^{n-1} \left(T_{\pi(i)} - \sum_{\ell=1}^{i} C_{\pi(\ell)}\right) U_{\pi(i)}. \quad (22)$$

A further manipulation of the RHS of (22) allows us to write

$$\begin{aligned}\sum_{i=1}^{n-1} r_i^{(j)} U_i &\le \sum_{i=1}^{n-1} \left(T_{\pi(i)} - C_{\pi(i)} - \sum_{\ell=1}^{i-1} C_{\pi(\ell)} \right) U_{\pi(i)} \\ &= \sum_{i=1}^{n-1} \left(T_{\pi(i)} - C_{\pi(i)} \right) U_{\pi(i)} - \sum_{i=1}^{n-1} \sum_{\ell=1}^{i-1} C_{\pi(\ell)} U_{\pi(i)} \\ &= \sum_{i=1}^{n-1} \left(T_i - C_i \right) U_i - \sum_{i=1}^{n-1} \sum_{\ell=1}^{i-1} T_{\pi(\ell)} U_{\pi(\ell)} U_{\pi(i)} \\ &\le \sum_{i=1}^{n-1} \left(T_i - C_i \right) U_i - \sum_{i=1}^{n-1} \sum_{\ell=1}^{i-1} \min\{T_i, T_\ell\} U_\ell U_i \\ &= \sum_{i=1}^{n-1} \left(T_i - C_i \right) U_i - \beta_n \end{aligned} \tag{23}$$

with β_n defined as in (20). The most delicate step above exploits the inequality

$$\sum_{i=1}^{n-1} \sum_{\ell=1}^{i-1} T_{\pi(\ell)} U_{\pi(\ell)} U_{\pi(i)} \ge \sum_{i=1}^{n-1} \sum_{\ell=1}^{i-1} \min\{T_i, T_\ell\} U_\ell U_i. \tag{24}$$

Such an inequality holds true for the following reasons:

- since π is a bijective self-map over $\{1, \ldots, n-1\}$, the double summation is made over all pairs of distinct indices of high priority tasks;
- the double summation can then be re-ordered and lower bounded if each term $U_\ell U_i$ is multiplied by the minimum among the two possible periods T_ℓ and T_i.

Also notice that in the inequality of (24), the equal sign holds when π is the permutation π^* which sorts the periods of the high priority tasks decreasingly.

From the upper bound of (23) and Eq. (13), the following upper bound to the convergence point $t_*(jC_n)$ holds

$$t_*(jC_n) \le \frac{jC_n + \sum_{i=1}^{n-1} C_i(1-U_i) - \beta_n}{1 - \sum_{i=1}^{n-1} U_i}, \tag{25}$$

for all $j \in \mathcal{J}^*$. From the definition of the response time of the n-th task of (7), the following upper bound follows

$$\begin{aligned} R_n &= \max_{j \in \mathcal{J}^*} \{ t_*(jC_n) - (j-1)T_n \} \\ &\le \max_{j \in \mathcal{J}^*} \left\{ \frac{jC_n + \sum_{i=1}^{n-1} C_i(1-U_i) - \beta_n}{1 - \sum_{i=1}^{n-1} U_i} - (j-1)T_n) \right\} \\ &= \frac{C_n + \sum_{i=1}^{n-1} C_i(1-U_i) - \beta_n}{1 - \sum_{i=1}^{n-1} U_i} = \hat{R}_n \end{aligned}$$

since the maximum is taken at $j = 1$. This concludes the proof. □

Notice that, for $n = 1$ or $n = 2$, we have $\beta_n = 0$, and the response time upper bound of (19) coincides then with the previously proposed one [16], [17], [18]. For $n \ge 3$, the response time upper bound of (19) is strictly tighter (smaller) than the upper bound of [16], [17], [18] by an amount

$$\frac{\beta_n}{1 - \sum_{i=1}^{n-1} U_i}.$$

If the tasks' priorities are Rate Monotonic (RM), then the next corollary provides a simplification of the response time bound of Theorem 1.

Corollary 2. Let the task set be schedulable by RM priorities. Then, the response time R_n of the n-th task is bounded from above as follows:

$$R_n \le \frac{C_n + \sum_{i=1}^{n-1} U_i \left(T_i - \sum_{j=1}^{i} C_j \right)}{1 - \sum_{i=1}^{n-1} U_i} \tag{26}$$

Proof: By the RM ordering of tasks, the value of β_n of (20) is

$$\begin{aligned} \beta_n &= \sum_{i=1}^{n-1} \sum_{j=1}^{i-1} \min\{T_i, T_j\} U_i U_j \\ &= \sum_{i=1}^{n-1} \sum_{j=1}^{i-1} T_j U_i U_j = \sum_{i=1}^{n-1} \sum_{j=1}^{i-1} C_j U_i. \end{aligned}$$

The statement of the corollary follows by replacing this expression for β_n in Equation (19). □

Notice that the complexity of computing the response time upper bound $\widehat{R}_n$ of (19) is quadratic in the number n of tasks in the set, as the term β_n requires to compute the product $U_i U_j$ for all pairs (i, j). The complexity of computing the response time upper bound for all tasks in the set remains quadratic, since the computation of any $\widehat{R}_i$ can take advantage of the β_{i-1} previously computed for $\widehat{R}_{i-1}$.

4. Evaluation of the tightness

In this section we evaluate the tightness of the response time upper bound $\widehat{R}_n$ of (19) by means of both exact results (in Section 4.1) and simulations (in Section 4.2). We mostly focus our attention to the full utilization case

$$\sum_{i=1}^{n} U_i = 1,$$

for the following reasons:

- as shown by Rothvoß and Eisenbrand [9], and also illustrated in Figure 1, the time complexity of RTA grows significantly as the total utilization approaches 1. Hence, understanding the tightness in this condition is particularly useful to overcome the problem of long run-time of RTA;
- high utilization tasks' sets are capable of better exploiting the available computing resource, leading to savings in processor speed and then in power.

First, we report some exact results in the special case of $n = 2$ tasks (Section 4.1). Although the case of two tasks has limited applicability, in Section 4.2 we show a numerical evaluation in which the same tightness result (proved for two tasks) could not be refuted.

4.1. The two tasks case: the bound is tight

When the number of tasks is two and the task set has full utilization, the response time $R_{2,j}$ of any job j of the 2nd task can be explicitly computed. We mention that Lemma 3 and Theorem 5 hold more generally in the case of real parameters, but we do not explore this direction in this article.

Lemma 3. Let $n = 2$ and $U_1 + U_2 = 1$. The response time $R_{2,j}$ of the j-th job of the 2nd task is given by

$$R_{2,j} = T_2 + \rho_j C_1, \tag{27}$$

where

$$\rho_j \coloneqq \left\lceil j\frac{T_2}{T_1} \right\rceil - j\frac{T_2}{T_1}\,. \tag{28}$$

Proof: To compute the response time of any job j of the second task, we take advantage of some results in [18]. The *worst-case workload* $W_1(t)$, that is the maximum amount of time over any contiguous interval of length t during which the 1st task is executing (Def. 1 in [18]), is

$$W_1(t) = \min\left\{ C_1 \left\lceil \frac{t}{T_1} \right\rceil, t - \left\lfloor \frac{t}{T_1} \right\rfloor (T_1 - C_1) \right\}.$$

The *worst-case idle time* $H_1(t)$ (Def. 2 in [18]), then simply is

$$H_1(t) = t - W_1(t).$$

Finally, by defining the *pseudo-inverse* $X_1(c)$ of $H_1(t)$, which is (Def. 3 in [18])

$$X_1(c) = \min_t\{t : H_1(t) \geq c\} = c + \left\lceil \frac{c}{T_1 - C_1} \right\rceil C_1$$

the finishing time $f_{2,j}$ of the j-th job of the 2nd task can be written as the largest instant at which the first task allows it to run for jC_2 amount of time, that is

$$f_{2,j} = X_1(jC_2) = jC_2 + \left\lceil \frac{jC_2}{T_1 - C_1} \right\rceil C_1$$

and then the response time of such a job is

$$R_{2,j} = jC_2 + \left\lceil \frac{jC_2}{T_1 - C_1} \right\rceil C_1 - (j-1)T_2.$$

With the additional hypothesis of $U_1 + U_2 = 1$, which implies $C_2 = (T_1 - C_1)\frac{T_2}{T_1}$, we find

$$\begin{aligned} R_{2,j} &= jC_2 + \left\lceil \frac{jC_2}{T_1 - C_1} \right\rceil C_1 - (j-1)T_2 \\ &= j(T_1 - C_1)\frac{T_2}{T_1} + \left\lceil j\frac{T_2}{T_1} \right\rceil C_1 - (j-1)T_2 \\ &= jT_2 - jC_1\frac{T_2}{T_1} + \left\lceil j\frac{T_2}{T_1} \right\rceil C_1 - jT_2 + T_2 \\ &= T_2 + \left(\left\lceil j\frac{T_2}{T_1} \right\rceil - j\frac{T_2}{T_1} \right) C_1 \end{aligned}$$

as required. □

Corollary 4. Let $n = 2$ and $U_1 + U_2 = 1$. The response time R_2 of the 2nd task is

$$R_2 = T_2 + \left(1 - \frac{1}{b}\right) C_1, \tag{29}$$

where b is the denominator of the fraction T_2/T_1 when written in lowest terms.

Proof: Let us write T_2/T_1 in lowest terms, that is, $T_2/T_1 = a/b$ where a and b are coprime positive integers. By Lemma 3 we have $R_{2,j} = T_2 + \rho_j C_1$ where $\rho_j = \left\lceil \frac{ja}{b} \right\rceil - \frac{ja}{b}$. Note the equality

$$\rho_j = \left\lceil \frac{\sigma_j}{b} \right\rceil - \frac{\sigma_j}{b}$$

where σ_j is the remainder of the division of ja by b, that is, $\sigma_j \coloneqq ja - \left\lfloor \frac{ja}{b} \right\rfloor b$. It is easy to see that σ_j is b-periodic, that is, $\sigma_{j+b} = \sigma_j$ for each positive integer j. Therefore ρ_j and, in turn, $R_{2,j}(x)$ are b-periodic as well. Thus the response time $R_2 = \max_{j\geq 1}(T_2 + \rho_j C_1)$ is found by maximizing ρ_j over the range $j = 1, \ldots, b$. For $j = b$ we have $\rho_b = 0$, so we concentrate on the cases $j = 1, \ldots, b-1$. In the integer sequence

$$\{a, 2a, 3a, \ldots, (b-1)a\}$$

no two numbers are congruent modulo b. In fact, if there were $s, t \in \mathbb{N}$ such that $1 \leq s < t < b$ and $ta = sa \mod b$, then b would divide $(t-s)a$, and since a and b are coprime it follows that b would divide $(t-s)$, which is impossible because $t - s < b$. This means that the remainders of the division by b of the numbers in the above sequence are all distinct, forming a permutation of $\{1, 2, \ldots, b-1\}$. Since for $j = 1, 2, \ldots, b-1$ we have $\left\lceil \frac{\sigma_j}{b} \right\rceil = 1$, then the maximum of ρ_j is attained when we have $\sigma_j = 1$, thus obtaining Equation (29), as required. □

Next we derive a strong tightness result in the special case of two tasks and full utilization.

Theorem 5. Given a set of two tasks with parameters C_1, T_1, C_2, T_2, let $R_2(C_1, T_1, C_2, T_2)$ denote the response time of the 2nd task, and $\widehat{R}_2(C_1, T_1, C_2, T_2)$ the response time upper bound of (19). Then $\widehat{R}_2(C_1, T_1, C_2, T_2)$ is the lowest upper bound to $R_2(C_1, T_1, C_2, T_2)$ which is continuous over the full-utilization sets of tasks

$$\mathcal{S} \coloneqq \left\{ (C_1, T_1, C_2, T_2) \in \mathbb{Q}^4_{>0} \;:\; \frac{C_1}{T_1} + \frac{C_2}{T_2} = 1 \right\}.$$

In other words, if $\widetilde{R}$ is a continuous function over S such that for each $x \in \mathcal{S}$ we have $R_2(x) \leq \widetilde{R}(x) \leq \widehat{R}_2(x)$, then $\widetilde{R}(x) = \widehat{R}_2(x)$ for each $x \in \mathcal{S}$.

Proof: First, we recall that if $n = 2$ then the response time upper bound of (19) is

$$\widehat{R}_2 = \frac{C_2}{1 - U_1} + C_1,$$

which, in the special case of full utilization $U_1 + U_2 = 1$, becomes

$$\widehat{R}_2 = T_2 + C_1.$$

Next, we recall the explicit formula for the response time of the second task, Equation (29):

$$R_2 = T_2 + \left(1 - \frac{1}{b}\right) C_1.$$

Now, let ϕ be a continuous function defined on $\mathcal{S}$ and assume there is $x = (C_1, T_1, C_2, T_2)$ such that $\phi(x) < \widehat{R}_2(x) = T_2 + C_1$. We want to show that there is $y \in \mathcal{S}$ such that $\phi(y) < R_2(y)$, thereby proving that ϕ is not an upper bound for R_2. To this end, let us consider an increasing sequence $n \mapsto p_n$ of prime numbers and let us perturb the task set x as follows:

$$x_n \coloneqq (C_{1,n}, T_{1,n}, C_2, T_2)$$

where

$$C_{1,n} \coloneqq C_1 \frac{p_n}{p_n + 1}\,, \quad T_{1,n} \coloneqq T_1 \frac{p_n}{p_n + 1}\,.$$

Note that $x_n \in \mathcal{S}$ for each n and that $x_n \to x$ as $n \to \infty$. Let us write

$$0 < \widehat{R}_2(x) - \phi(x) = A_n + B_n + C_n$$

where $A_n \coloneqq \widehat{R}_2(x) - R_2(x_n)$, $B_n \coloneqq R_2(x_n) - \phi(x_n)$, $C_n \coloneqq \phi(x_n) - \phi(x)$ and let us consider these three n-dependent terms separately. The denominator of $T_2/T_{1,n}$ (in lowest terms) diverges as $n \to \infty$, so the response time of the second task of x_n, as given in Equation (29), tends to $T_2 + C_1$, implying $A_n \to 0$ as $n \to \infty$. Moreover, we have $C_n \to 0$ as $n \to \infty$ because ϕ is continuous. Therefore, $B_n \to (\widehat{R}_2(x) - \phi(x))$ as $n \to \infty$ and this implies that, for a sufficiently large $\bar{n}$, we have $0 < \frac{1}{2}(\widehat{R}_2(x) - \phi(x)) < B_{\bar{n}}$. In other words, setting $y = x_{\bar{n}}$ we have $\phi(y) < R_2(y)$, as required. □

4.2. Tightness in the general case

Proving an equivalent version of Theorem 5 for the general case of any number n of tasks was unfortunately beyond our capabilities. To investigate whether or not an equivalent version of Theorem 5 may hold for any arbitrary number of tasks we performed several experiments.

In the experiment illustrated in Figure 4, we set:

- $C_1 = 1$, $T_1 = 5$, $C_3 = 2$, $T_3 = 6$;
- T_2 varies along the x-axis from 4 to 7 with step equal to $1/60$, U_2 is set constantly to $\frac{1}{4}$ and C_2 is calculated correspondingly as $C_2 = T_2 \times U_2$;
- $T_4 = 10$ always, and C_4 is calculated to achieve the full utilization condition that is

$$C_4 = T_4 \left(1 - \sum_{i=1}^{3} \frac{C_i}{T_i}\right).$$

In the figure, we report the response time upper bound R_4^{ub} previously proposed by several authors [16], [17], [18] (dashed line), the new response time upper bound $\widehat{R}_4$ proposed in this paper (solid line), and the exact response time R_4 (black dots) computed with standard RTA methods [1],

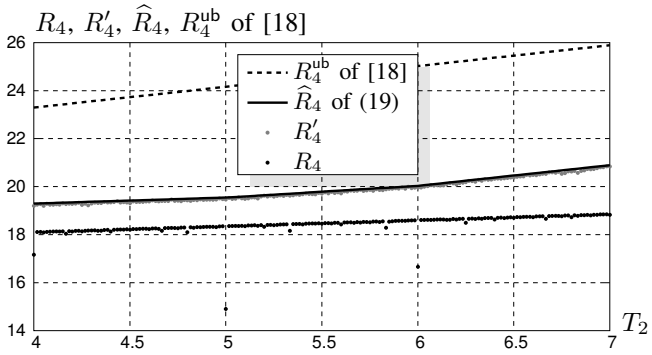

Figure 4. Response time R_4 as T_2 varies.

[3]. It can be observed that the exact response time R_4 seems to have a continuous upper bound considerably lower than our proposed bound (solid black line), with a non-negligible gap between such a smooth upper limit and the response time upper bound $\widehat{R}_4$ as it spans from about 1.2 (when $T_2 = 4$) to 2 (when $T_2 = 7$). We believe, however, that such a gap depends on the harmonicity of periods, as demonstrated in Corollary 4 for the 2-tasks case. In fact, the gap between R_4 and $\widehat{R}_4$ widens whenever T_2 is such that the hyperperiod of the task periods (that is $\mathsf{lcm}(T_1, T_2, T_3, T_4)$) gets smaller. This can be clearly observed when $T_2 = 5$ (which is equal to T_1) or $T_2 = 6$ (equal to T_3). In such a harmonic case the response time R_4 is noticeably lower than the response time upper bound $\widehat{R}_4$. To evaluate the dependency on the harmonicity of the periods, we performed a new simulation with task periods close to the previous ones, but with extremely poor harmonicity. We chose:

- $T'_1 = 5 + 10^{-3} \times \pi$,
- T_2 still spanning from 4 to 7 with steps of $1/60$,
- $T'_3 = 6 + 10^{-3} \times \sqrt{2}$, and
- $T'_4 = 10 + 10^{-3} \times \sqrt{3}$.
- The computation times are the same as in the previous experiment, that is: $C_1 = 1$, $C_2 = 0.25 \times T_2$, $C_3 = 2$, and C_4 calculated such that the total utilization is 1.

We remark that, since irrational numbers cannot be finitely represented in floating point arithmetic, in our simulation the periods were assigned to the closest finite floating point representation over 64 bits (size of `double`). Also, these values of periods and the full utilization hypothesis imply that the busy period is unbounded. Hence, the computation of the RTA for R'_4 had to be stopped at some time horizon. In this simulation, we stopped the RTA at time 10.000.000.

In Figure 4, the response time R'_4 of the fourth task of such a perturbed task set is drawn in gray dots. Such a small perturbation (of the order of 10^{-3}) in the task periods makes the response time R'_4 jump extremely close to the proposed upper bound $\widehat{R}_4$. Based on this experiment, and many others which provide the same indication but cannot be reported here due to space limits, we conjecture that a tightness result (analogous to Theorem 5 for two tasks) may hold for any number n of tasks in the set.

5. Extensions

The RTA has been extended from the basic Liu and Layland task model to cover a broader set of application models. Similarly to the work by Davis and Burns [17], we aim here to extend the response time upper bound of Theorem 1 to a richer task model, which includes:

1) the task *jitter* J_i, which represents a range of variability of the task arrivals around the ideal periodic case [3];
2) the task *blocking time* B_i, which is the longest time the i-th task may be blocked by lower priority tasks (due, for example, to the usage of shared resources);
3) the *cache-related preemption delay* $\gamma_{n,i}$, representing the preemption cost due to each job of the i-th high priority task executing within the worst-case response time of the n-th task [28], [29].

First, in Corollary 6 we extend the response time upper bound to the case of tasks with jitter only. Later, in Corollary 7 we account for the more complete case with blocking time and cache-related preemption delay, as well.

Corollary 6. Let J_i be the jitter of the i-th task. Then the response time R_n of the n-th task is upper bounded by

$$\widehat{R}_n := \frac{C_n + \sum_{i=1}^{n-1} J_i\, U_i + \sum_{i=1}^{n-1} (T_i - C_i) U_i - \beta_n}{1 - \sum_{i=1}^{n-1} U_i} + J_n, \tag{30}$$

with β_n defined as in (20).

Proof: Let $t_*(c)$ denote the convergence point of (5) with $g(c,t)$ now accounting for the task jitter, as follows

$$g(c,t) = c + \sum_{i=1}^{n-1} \left\lceil \frac{t + J_i}{T_i} \right\rceil C_i. \tag{31}$$

As shown by Tindell et al. [26], in this case with task jitter, the response time R_n of the n-th task is

$$R_n = \max_{j \in \mathcal{J}^*} \{ t_*(jC_n) - ((j-1)T_n - J_n) \}. \tag{32}$$

with $\mathcal{J}^*$ being redefined as

$$\mathcal{J}^* = \{ j \in \mathbb{N} \ : \ j \geq 1,\ t_*((j-1)C_n) \geq (j-1)T_n - J_n \}. \tag{33}$$

Following the same steps of the proof of Theorem 1, let $q_i^{(j)}$ and $r_i^{(j)}$ be quotient and remainder of the Euclidean division of $t_*(jC_n) + J_i$ by T_i, that is

$$\begin{aligned} q_i^{(j)} &:= \left\lceil \frac{t_*(jC_n) + J_i}{T_i} \right\rceil \geq 1 \\ r_i^{(j)} &:= q_i^{(j)} T_i - (t_*(jC_n) + J_i). \end{aligned} \tag{34}$$

From (34), by following analogous steps that lead to (13), we can find the expression of the fixed point $t_*(jC_n)$ as a function of the remainders $r_i^{(j)}$ of the Euclidean division, which is

$$t_*(jC_n) = \frac{jC_n + \sum_{i=1}^{n-1} J_i\, U_i + \sum_{i=1}^{n-1} r_i^{(j)} U_i}{1 - \sum_{i=1}^{n-1} U_i}.$$

The quantity $\sum_{i=1}^{n-1} r_i^{(j)} U_i$ can then be upper bounded exactly in the same way as in Eq. (23) of the proof of Theorem 1, which leads to the following bound to the fixed point $t_*(jC_n)$

$$t_*(jC_n) \leq \frac{jC_n + \sum_{i=1}^{n-1} J_i\, U_i + \sum_{i=1}^{n-1} (T_i - C_i) U_i - \beta_n}{1 - \sum_{i=1}^{n-1} U_i},$$

with β_n as in (20). From the definition of the response time R_n of (32) in the case with jitter, the bound to the response time follows. □

The extension of the proposed response time upper bound also to the case with blocking time and cache related preemption delay is straightforward, as reported in the next Corollary.

Corollary 7. Let:

- J_i be the release jitter of the i-th task;
- B_n be the blocking time of the n-th task; and
- $\gamma_{n,i}$ be the cost of a preemption by the i-th task to the n-th.

Then, the response time can be upper bounded by

$$\widehat{R}_n := \frac{C'_n + \sum_{i=1}^{n-1} J_i\, U'_i + \sum_{i=1}^{n-1} (T_i - C'_i) U'_i - \beta'_n}{1 - \sum_{i=1}^{n-1} U'_i} + J_n, \tag{35}$$

with:

$$\begin{aligned} C'_n &= C_n + B_n \\ C'_i &= C_i + \gamma_{n,i}, \qquad U'_i = \frac{C'_i}{T_i}, \end{aligned}$$

and β'_n defined as in (20), with the modified utilizations U'_i.

Proof: With this hypothesis, the exact response time is defined (Eq. (12) in [29]) as the fixed point of

$$R_n = C_n + B_n + \sum_{i=1}^{n-1} \left\lceil \frac{R_n + J_i}{T_i} \right\rceil (C_i + \gamma_{n_i}).$$

The corollary then follows directly by replacing C_i with C'_i for all $i = 1, \ldots, n$ in this definition and then invoking Corollary 6. □

6. Conclusions and future works

In this paper, we have proposed an upper bound to the response time of tasks with arbitrary deadline to be scheduled by fixed priorities. The proposed bound is extended to the case with task blocking time, release jitter, and cache-related preemption delay. Due to its quadratic time complexity, such a bound can substitute the exact response time in all analysis techniques which require an extensive computation of RTA (such as in the holistic analysis [26], [8]). Due to its continuity, it can be effectively used in design optimization techniques [24] as the suppression of the integer variables modeling the number of interferences can significantly improve the performance of solvers.

The tightness of the bound is then evaluated. First it is proved that, under simplifying assumptions of only two tasks in the set, the proposed bound is the tightest continuous function upper bounding the exact response time. Simulations suggest that the same property may also hold in more general cases. However, the formal proof of this conjecture or the search of any counter-example is left as future work.

References

[1] M. Joseph and P. K. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, Oct. 1986.

[2] P. K. Harter, "Response times in level-structured systems," *ACM Transactions on Computer Systems*, vol. 5, no. 3, pp. 232–248, Aug. 1987.

[3] N. C. Audsley, A. Burns, M. Richardson, K. W. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority preemptive scheduling," *Software Engineering Journal*, vol. 8, no. 5, pp. 284–292, Sep. 1993.

[4] S. Baruah and E. Bini, "Partitioned scheduling of sporadic task systems: an ILP-based approach," in *Conference on Design and Architectures for Signal and Image Processing*, Bruxelles, Belgium, Nov. 2008.

[5] B. Andersson and J. Jonsson, "Some insights on fixed-priority preemptive non-partitioned multiprocessor scheduling," Dept. of Computer Engineering, Chalmers University of Technology, Tech. Rep., Apr. 2000.

[6] M. Bertogna and M. Cirinei, "Response-time analysis for globally scheduled symmetric multiprocessor platforms," in *Proceedings of the 28th IEEE Real-Time Systems Symposium*, Tucson, Arizona, 2007, pp. 149–160.

[7] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocessing and Microprogramming*, vol. 50, pp. 117–134, Apr. 1994.

[8] J. C. Palencia Gutiérrez, J. J. Gutiérrez García, and M. González Harbour, "On the schedulability analysis for distributed hard real-time systems," in *Proceedings of the 9th Euromicro Workshop on Real-Time Systems*, Toledo, Spain, Apr. 1997, pp. 136–143.

[9] F. Eisenbrand and T. Rothvoß, "Static-priority real-time scheduling: Response time computation is NP-hard." in *Proceedings of the 29th IEEE Real-Time Systems Symposium*, Barcelona, Spain, Nov. 2008, pp. 397–406.

[10] J. P. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadline," in *Proceedings of the* 11^{th} *IEEE Real-Time Systems Symposium*, Lake Buena Vista (FL), U.S.A., Dec. 1990, pp. 201–209.

[11] V. Bonifaci, A. Marchetti-Spaccamela, N. Megow, and A. Wiese, "Polynomial-time exact schedulability tests for harmonic real-time tasks," in *Proceedings of the 34th IEEE Real-Time Systems Symposium*, Vancouver, BC, Canada, Dec. 2013, pp. 236–245.

[12] K. Albers and F. Slomka, "An event stream driven approximation for the analysis of real-time systems," in *Proceedings of the* 16^{th} *Euromicro Conference on Real-Time Systems*, Catania, Italy, Jun. 2004, pp. 187–195.

[13] N. Fisher and S. Baruah, "A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems," in *Proceedings of the* 17^{th} *Euromicro Conference on Real-Time Systems*, Palma de Mallorca, Spain, Jul. 2005, pp. 117–126.

[14] T. H. C. Nguyen, P. Richard, and N. Fisher, "The fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with arbitrary relative deadlines revisited," in *18th International Conference on Real-Time and Network Systems*, 2010, pp. 21–30.

[15] T. H. C. Nguyen, P. Richard, and E. Grolleau, "An FPTAS for response time analysis of fixed priority real-time tasks with resource augmentation," *IEEE Transactions on Computers*, vol. 64, no. 7, pp. 1805–1818, Jul. 2015.

[16] E. Bini and S. K. Baruah, "Efficient computation of response time bounds under fixed-priority scheduling," in *Proceedings of the* 15^{th} *conference on Real-Time and Network Systems*, Nancy, France, Mar. 2007, pp. 95–104.

[17] R. I. Davis and A. Burns, "Response time upper bounds for fixed priority real-time systems," in *Proceedings of the 29th IEEE Real-Time Systems Symposium*, Barcelona, Spain, Dec. 2008, pp. 407–418.

[18] E. Bini, T. H. C. Nguyen, P. Richard, and S. K. Baruah, "A response-time bound in fixed-priority scheduling with arbitrary deadlines," *IEEE Transactions on Computers*, vol. 58, no. 2, pp. 279–286, Feb. 2009.

[19] E. Bini, "The quadratic utilization upper bound for arbitrary deadline real-time tasks," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 593–599, Feb. 2015.

[20] C. G. Okwudire, M. M. van den Heuvel, R. J. Bril, and J. J. Lukkien, "Exploiting harmonic periods to improve linearly approximated response-time upper bounds," in *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation*, Bilbao, Spain, Sep. 2010, pp. 1–4.

[21] J.-J. Chen, W.-H. Huang, and C. Liu, "k2Q: A quadratic-form response time and schedulability analysis framework for utilization-based analysis," Cornell University Library, Tech. Rep. arXiv: 1505.03883, 2015, available at http://arxiv.org/abs/1505.03883.

[22] S. Baruah, "Efficient computation of response time bounds for preemptive uniprocessor deadline monotonic scheduling," *Real-Time Systems*, vol. 47, pp. 517–533, 2011.

[23] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the Association for Computing Machinery*, vol. 20, no. 1, pp. 46–61, Jan. 1973.

[24] Q. Zhu, H. Zeng, W. Zheng, M. Di Natale, and A. Sangiovanni-Vincentelli, "Optimization of task allocation and priority assignment in hard real-time distributed systems," *ACM Transactions on Embedded Computing Systems*, vol. 11, no. 4, pp. 85:1–85:30, Jan. 2013.

[25] J.-E. Kim, T. Abdelzaher, and L. Sha, "Budgeted generalized rate monotonic analysis for the partitioned, yet globally scheduled uniprocessor model," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015 IEEE*, 2015, pp. 221–231.

[26] K. W. Tindell, A. Burns, and A. Wellings, "An extendible approach for analysing fixed priority hard real-time tasks," *Journal of Real Time Systems*, vol. 6, no. 2, pp. 133–152, Mar. 1994.

[27] M. Sjödin and H. Hansson, "Improved response-time analysis calculations," in *Proceedings of the* 19^{th} *IEEE Real-Time Systems Symposium*, Madrid, Spain, Dec. 1998, pp. 399–408.

[28] J. V. Busquets-Mataix, J. J. Serrano, R. Ors, P. Gil, and A. Wellings, "Adding instruction cache effect to schedulability analysis of preemptive real-time systems," in *Proceedings of the Real-Time Technology and Applications Symposium*, 1996, pp. 204–212.

[29] S. Altmeyer, R. I. Davis, and C. Maiza, "Cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems," in *Proceedings of the 32nd IEEE Real-Time Systems Symposium*. IEEE, 2011, pp. 261–271.