Tight Tardiness Bounds for Pseudo-Harmonic Tasks under Global-EDF-Like Schedulers

Shareef Ahmed
University of North Carolina at Chapel Hill, , USA

James H. Anderson
University of North Carolina at Chapel Hill, USA

Abstract

The global earliest-deadline-first (GEDF) scheduler and its variants are soft-real-time (SRT) optimal for periodic/sporadic tasks, meaning they provide bounded tardiness so long as the underlying platform is not over-utilized. Although their SRT-optimality has long been known, tight tardiness bounds for these schedulers have remained elusive. In this paper, a tardiness bound, that does not depend on the processor or task count, is derived for pseudo-harmonic periodic tasks, which are commonly used in practice, under global-EDF-like (GEL) schedulers. This class of schedulers includes both GEDF and first-in-first-out (FIFO). This bound is shown to be generally tight via an example. Furthermore, it is shown that exact tardiness bounds for GEL-scheduled pseudo-harmonic periodic tasks can be computed in pseudo-polynomial time.

2012 ACM Subject Classification Computer systems organization → Real-time systems

Keywords and phrases soft real-time systems, multicore, tardiness bounds

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2021.11

Funding Work was supported by NSF grants CNS 1563845, CNS 1717589, CPS 1837337, CPS 2038855, and CPS 2038960, ARO grant W911NF-20-1-0237, and ONR grant N00014-20-1-2698.

1 Introduction

The rise of multicore platforms has generated much interest in global schedulers such as the global earliest-deadline-first (GEDF) scheduler. Although the preemptive uniprocessor earliest-deadline-first (EDF) scheduler is *hard real-time (HRT) optimal*, meaning it can schedule any task system that does not over-utilize the underlying platform without any deadline misses, preemptive GEDF[1] is not HRT-optimal [10]. Despite this, GEDF and many of its variants guarantee bounded tardiness on different types of multiprocessor platforms for any task system that does not over-utilize the platform [9, 20, 26, 28], making them *soft real-time (SRT) optimal*. The significance of GEDF's SRT-optimality is reflected by references to it in the documentation of `SCHED_DEADLINE` [13], Linux's GEDF implementation.

Unfortunately, all known tardiness bounds for GEDF and its variants increase with respect to the number of processors. Moreover, experimental evaluations have shown that these bounds tend to become looser as the processor count increases [27]. This causes the corresponding SRT guarantees to be of questionable utility on large platforms and may even increase system cost. For example, the ill effects of tardiness can be "hidden" by buffering [11, 16] and the needed buffers must be sized based upon established tardiness bounds. While HRT-optimal schedulers can ameliorate these problems by eliminating all tardiness, they come at the expense of large overheads [2, 4, 5, 23]. Hence, a tardiness bound that does not scale with the number of tasks or processors under practical global schedulers

[1] All schedulers mentioned herein are assumed to be preemptive unless noted otherwise.

like GEDF would be desirable; the derivation of such a bound has remained an open problem since the first work on SRT-optimality in 2005 [8].

In this paper, we close this problem for an important category of task systems, namely *pseudo-harmonic periodic task systems*, where every period divides the maximum period. In particular, we establish a tight tardiness bound for pseudo-harmonic periodic tasks under *global-EDF-like* (GEL) schedulers on identical multiprocessor platforms. Our tardiness bound does not depend on the processor or task count, but scales with respect to the task parameters, e.g, periods. The class of GEL schedulers includes not only GEDF, but first-in-first-out (FIFO) and various other related schedulers. Pseudo-harmonic tasks are common in automotive applications [17]. Moreover, the class of pseudo-harmonic task systems contains *harmonic task systems*, where every period is an integer multiple of each smaller period. Harmonic task systems are common in different application domains such as avionics, robotics, control applications, etc. [3, 6, 14, 21, 24]. To our knowledge, we are the first to establish a tardiness bound that is tight in general for a class of task systems of practical interest under a job-level fixed-priority global scheduler. Our work was inspired by prior seminal work on the periodic behavior of GEDF schedules for HRT periodic systems [7, 15, 22].

Prior work. The SRT-optimality of GEDF on identical multiprocessor platforms was first shown by Devi and Anderson [9]. A tighter tardiness bound under GEDF can be obtained by *compliant vector analysis* (CVA), proposed by Erickson et. al [11,12]. The current best-known GEDF tardiness bound, the *harmonic bound*, was given by Valente [27]. *Window-constrained* schedulers, a class of schedulers containing all GEL schedulers, were proven to be SRT-optimal by Leontyev and Anderson [20]. Recent works have established the SRT-optimality of both GEDF on uniform heterogeneous multiprocessor platforms, and window-constrained schedulers on identical multiprocessor platforms with arbitrary affinity masks [25, 28].

Contributions. Our contributions are four-fold. First, we give a tardiness bound that is independent of the task or processor count for pseudo-harmonic periodic tasks under GEL schedulers. In a GEL scheduler, each task has a task-level fixed parameter called its *relative priority point*, which is used to assign a *priority point (PP)* to each of its jobs: the PP of a job is determined by adding its task's relative PP to the job's release time. The priority of a job is determined by its PP, with earlier PPs denoting higher priority. For example, under GEDF (resp., FIFO), a job's PP is given by its deadline (resp., release time). Additionally, we show that our bound can be exploited to ensure tardiness bounds that do not depend on the processor count for pseudo-harmonic sporadic tasks by using periodic servers scheduled by a GEL scheduler. Second, we show the general tightness of our bound by an example. Third, we give an upper bound on the length of the interval that needs to be simulated to derive an exact tardiness bound of any task in a pseudo-harmonic periodic task system. Using this, we show how to determine exact tardiness bounds in pseudo-polynomial time. To our knowledge, this is the first work on GEL schedulers that shows how to bound tardiness exactly. Fourth, we compare both of our bounds with each other and prior bounds by simulation experiments.

Organization. In the rest of this paper, we give necessary background information (Sec. 2), derive a tight tardiness bound for GEL schedulers (Sec. 3), show how to determine exact tardiness bounds in pseudo-polynomial time via schedule simulation (Sec. 4), discuss our experimental results (Sec. 5), and conclude (Sec. 6).

2 Preliminaries

We consider a task system τ consisting of n implicit-deadline periodic *tasks* $\tau_1, \tau_2, \ldots, \tau_n$ to be scheduled on m identical processors. Each task τ_i releases a potentially infinite sequence

of *jobs* $\tau_{i,1}, \tau_{i,2}, \ldots$. The *period* of task τ_i, denoted by T_i, is the separation time between two consecutive job releases by it. The largest period among all tasks is denoted by T_{max}. A task system is called *sporadic* when the separation time between consecutive jobs of each task τ_i can be more than T_i. The *worst-case execution cost* of τ_i is denoted by C_i. The *offset* of a periodic task τ_i, denoted by Φ_i, is the release time of $\tau_{i,1}$. The *relative deadline* of τ_i is $D_i = T_i$. For brevity, we denote a periodic task τ_i by (Φ_i, C_i, T_i).

The *release time*, *absolute deadline*, *completion time*, and *execution cost* of job $\tau_{i,k}$ are denoted by $r_{i,k}, d_{i,k}, f_{i,k}$, and $C_{i,k}$, respectively. The jobs of each task are sequential, i.e., $\tau_{i,k+1}$ cannot start execution before $\tau_{i,k}$ completes. The *tardiness* of a job $\tau_{i,k}$ is defined as $\max\{0, f_{i,k} - d_{i,k}\}$. The tardiness of task τ_i is the maximum tardiness among any of its jobs.

The *utilization* of τ_i is $u_i = C_i/T_i$. The *utilization* of the task system τ is $U = \sum_{i=1}^{n} u_i$. We require $u_i \leq 1.0$ and $U \leq m$ to hold; both are necessary for bounded tardiness [9]. The *hyperperiod* H is the least common multiple of all periods. The periods are *pseudo-harmonic* when each period divides T_{max}, i.e., $H = T_{max}$ holds.

The relative PP of a task τ_i is denoted by Y_i. We assume $Y_i \geq 0$ holds for each task τ_i. The maximum and minimum relative PP among all tasks in τ are denoted by Y_{max} and Y_{min}, respectively. The *priority point (PP)* of a job $\tau_{i,k}$, denoted by $y_{i,k}$, is defined as

$$y_{i,k} = r_{i,k} + Y_i. \tag{1}$$

If $y_{i,k} < y_{j,\ell}$, then job $\tau_{i,k}$ has higher priority than job $\tau_{j,\ell}$. We assume ties to be broken arbitrarily but consistently by task index.

We assume time to be discrete and a unit of time to be 1.0. All scheduling decisions are taken at integer points in time. We also assume all task parameters to be integers. Therefore, when a task τ_i executes during an unit interval $[t-1, t)$, it means τ_i continuously executes during $[t-1, t)$. A job completes execution at t if it executes for the last time during $[t-1, t)$. A job completes execution before t if it completes at or before $t-1$. (It can be shown that the tardiness bound presented in Sec. 3 also holds when time is continuous.) The following definitions closely follow from material in [1, 9, 28].

▶ **Definition 1.** *A job $\tau_{i,k}$ is* active *at time t in a schedule $\mathcal{S}$ if $r_{i,k} \leq t < d_{i,k}$.*

▶ **Definition 2.** *A job $\tau_{i,k}$ is* pending *at time t in a schedule $\mathcal{S}$ if $r_{i,k} \leq t$ and $\tau_{i,k}$ has not completed execution at or before t in S.*

Allocation. The cumulative processor capacity allocated to a task τ_i (resp., task system τ) in a schedule $\mathcal{S}$ over an interval $[t, t')$ is denoted by $\mathsf{A}_i(t, t', \mathcal{S})$ (resp., $\mathsf{A}(t, t', \mathcal{S})$). Thus, $\mathsf{A}(t, t', \mathcal{S}) = \sum_{\tau_i \in \tau} \mathsf{A}_i(t, t', \mathcal{S})$.

Ideal schedule. Let $\widehat{\pi}_1, \widehat{\pi}_2, \ldots, \widehat{\pi}_n$ be n processors with speeds $u_1, u_2, \ldots, u_n$, respectively. In an *ideal schedule* $\mathcal{I}$, each task τ_i is partitioned to execute on processor $\widehat{\pi}_i$. Each job starts execution as soon as it is released and completes execution by its deadline in $\mathcal{I}$. For task τ_i (resp., task system τ), $\mathsf{A}_i(t, t', \mathcal{I}) \leq u_i(t' - t)$ (resp., $\mathsf{A}(t, t, \mathcal{I}) \leq U(t' - t)$) holds. If τ_i is periodic and each job executes for its worst case C_i, then $\mathsf{A}_i(t, t', \mathcal{I}) = u_i(t' - t)$ where $t, t' \geq \Phi_i$.

lag and LAG. The lag of a task τ_i at time t in a schedule $\mathcal{S}$ is defined as

$$\mathsf{lag}_i(t, \mathcal{S}) = \mathsf{A}_i(0, t, \mathcal{I}) - \mathsf{A}_i(0, t, \mathcal{S}). \tag{2}$$

Since $\mathsf{lag}_i(0, \mathcal{S}) = 0$, for $t' \geq t$ we have

$$\mathsf{lag}_i(t', \mathcal{S}) = \mathsf{lag}_i(t, \mathcal{S}) + \mathsf{A}_i(t, t', \mathcal{I}) - \mathsf{A}_i(t, t', \mathcal{S}). \tag{3}$$

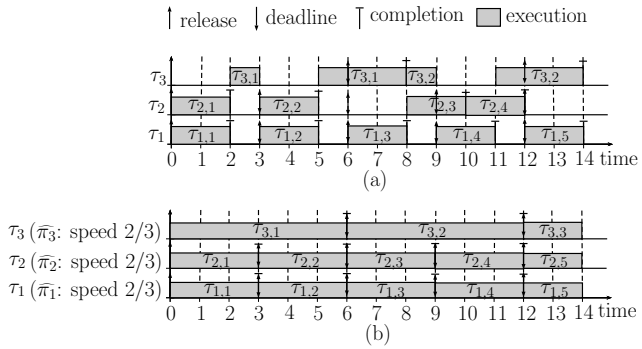


Figure 1 (a) A GEDF schedule and (b) an ideal schedule of the task system in Ex. 3.

The LAG of a task system τ in a schedule $\mathcal{S}$ at time t is defined as

$$\mathsf{LAG}(t, \mathcal{S}) = \sum_{\tau_i \in \tau} \mathsf{lag}_i(t, \mathcal{S}) = \mathsf{A}(0, t, \mathcal{I}) - \mathsf{A}(0, t, \mathcal{S}). \tag{4}$$

Since $\mathsf{LAG}(0, \mathcal{S}) = 0$, for $t' \geq t$ we have

$$\mathsf{LAG}(t', \mathcal{S}) = \mathsf{LAG}(t, \mathcal{S}) + \mathsf{A}(t, t', \mathcal{I}) - \mathsf{A}(t, t', \mathcal{S}). \tag{5}$$

▶ **Example 3.** Consider a periodic task system τ with tasks $\tau_1 = (0, 2, 3), \tau_2 = (0, 2, 3)$, and $\tau_3 = (0, 4, 6)$. A GEDF schedule $\mathcal{S}$ and an ideal schedule $\mathcal{I}$ of the task system is shown in insets (a) and (b) of Fig. 1, respectively. τ_1's allocation over interval $[0, 5)$ in $\mathcal{S}$ and $\mathcal{I}$ are 4.0 and 10/3 execution units, respectively. τ_1's lag in $\mathcal{S}$ at time 5 is $\mathsf{lag}_1(5, \mathcal{S}) = 10/3 - 4 = -2/3$. The LAG of the task system τ at time 5 is 1.0. ◀

3 Tardiness Bound

In this section, we derive a tardiness bound for pseudo-harmonic periodic task systems under a GEL scheduler. We assume $n > m$ as each job meets its deadline otherwise. We initially assume the following, which we relax later.

(B) Each job of any task τ_i executes for its worst-case execution cost C_i.

We consider a GEL schedule $\mathcal{S}$ of τ satisfying (B) to derive our tardiness bound. We derive our tardiness bound (Theorem 28) by giving an upper bound on per-task lag (Lemma 27) using a lag-*monotonicity* property (Lemma 17). Informally, the lag-monotonicity property states that no task τ_i receives more allocation in $\mathcal{S}$ than $\mathcal{I}$, i.e., lag does not decrease, over any interval of length T_{max} beginning at or after Φ_i. We first establish the lag-monotonicity property using a series of properties of lag proved in Sec. 3.1. We then use the lag-monotonicity property to derive our tardiness bound in Sec. 3.2.

3.1 lag Properties

We begin by proving some properties of lag. All properties specified here also hold for non-pseudo-harmonic periodic task systems satisfying (B) with T_{max} replaced by H in the properties that reference T_{max}. Since the lag-monotonicity property compares lag values between two time instants, we first establish several properties concering such comparisons

between a pair of lag values (Lemmas 11–15) based on the simpler properties of lag (Lemmas 4–10). Readers familiar with the concept of lag may skip the proofs of Lemmas 4–10.

▶ **Lemma 4.** *For any task τ_i and interval $[t, t')$ with $t \geq \Phi_i$, the following hold.*
(a) If τ_i continuously executes during $[t, t')$ in $\mathcal{S}$, then $\mathsf{lag}_i(t', \mathcal{S}) = \mathsf{lag}_i(t, \mathcal{S}) + (t' - t)(u_i - 1)$.
(b) If τ_i does not execute during $[t, t')$ in $\mathcal{S}$, then $\mathsf{lag}_i(t', \mathcal{S}) = \mathsf{lag}_i(t, \mathcal{S}) + (t' - t)u_i$.

Proof. Since $t \geq \Phi_i$, by the definition of $\mathcal{I}$, we have $\mathsf{A}_i(t, t', \mathcal{I}) = (t' - t)u_i$.
(a) Since τ_i continuously executes throughout $[t, t')$ in $\mathcal{S}$, $\mathsf{A}_i(t, t', \mathcal{S}) = (t' - t)$ holds. Substituting $\mathsf{A}_i(t, t', \mathcal{I})$ and $\mathsf{A}_i(t, t', \mathcal{S})$ in (3), we have $\mathsf{lag}_i(t', \mathcal{S}) = \mathsf{lag}_i(t, \mathcal{S}) + (t' - t)u_i - (t' - t) = \mathsf{lag}_i(t, \mathcal{S}) + (t' - t)(u_i - 1)$.
(b) Since τ_i does not execute during $[t, t')$ in $\mathcal{S}$, we have $\mathsf{A}_i(t, t', \mathcal{S}) = 0$. Substituting $\mathsf{A}_i(t, t', \mathcal{I})$ and $\mathsf{A}_i(t, t', \mathcal{S})$ in (3), we have $\mathsf{lag}_i(t', \mathcal{S}) = \mathsf{lag}_i(t, \mathcal{S}) + (t' - t)u_i - 0 = \mathsf{lag}_i(t, \mathcal{S}) + (t' - t)u_i$. ◀

▶ **Lemma 5** ([28])**.** *If $\mathsf{lag}_i(t, \mathcal{S}) > 0$, then τ_i has a pending job at t in $\mathcal{S}$.*

The following lemma states that the lag of any task τ_i is non-negative in $\mathcal{S}$ at any time instant t when it releases a job. Intuitively, all of τ_i's jobs released before t complete execution in $\mathcal{I}$ by time t, and thus, τ_i cannot receive more allocation in $\mathcal{S}$ than $\mathcal{I}$.

▶ **Lemma 6.** *For any task τ_i and non-negative integer c, $\mathsf{lag}_i(\Phi_i + cT_i, \mathcal{S}) \geq 0$.*

Proof. If $c = 0$, then the lemma trivially holds. Assume that there is a task τ_i and an integer $c \geq 1$ such that $\mathsf{lag}_i(\Phi_i + cT_i, \mathcal{S}) < 0$ holds. Then, by (2), $\mathsf{A}_i(0, \Phi_i + cT_i, \mathcal{S}) > \mathsf{A}_i(0, \Phi_i + cT_i, \mathcal{I})$ holds. Since τ_i releases periodically, $\Phi_i + cT_i$ is the deadline (resp., release time) of $\tau_{i,c-1}$ (resp., $\tau_{i,c}$). By the definition of $\mathcal{I}$, all jobs of τ_i released before $\Phi_i + cT_i$ complete execution by time $\Phi_i + cT_i$ in $\mathcal{I}$. Since no job can execute before its release, $\mathsf{A}_i(0, \Phi_i + cT_i, \mathcal{S})$ cannot be larger than $\mathsf{A}_i(0, \Phi_i + cT_i, \mathcal{I})$, a contradiction. ◀

Lemmas 7–10 give relationships among a task τ_i's lag at time t, its utilization, and the deadline or release time of a job of τ_i. We prove these lemmas by expressing τ_i's allocation in $\mathcal{S}$ by time t in terms of τ_i's utilization and the deadline or release time of a job of τ_i.

▶ **Lemma 7.** *If τ_i has no pending job at time $t \geq \Phi_i$ in $\mathcal{S}$ and $\tau_{i,k}$ is the active job of τ_i at t, then $\mathsf{lag}_i(t, \mathcal{S}) = (t - d_{i,k})u_i$.*

Proof. Since $\tau_{i,k}$ completes execution at or before t, all jobs of τ_i released at or before $r_{i,k}$ complete execution at or before t. Since $t < d_{i,k}$, no jobs released after $r_{i,k}$ execute before t. Hence, $\mathsf{A}_i(0, t, \mathcal{S}) = \sum_{j=1}^{k} C_i = \sum_{j=1}^{k} T_i u_i = \sum_{j=1}^{k} (r_{i,j+1} - r_{i,j})u_i = (r_{i,k+1} - r_{i,1})u_i = (d_{i,k} - \Phi_i)u_i$. By the definition of $\mathcal{I}$, we have $\mathsf{A}_i(0, t, \mathcal{I}) = (t - \Phi_i)u_i$. Substituting $\mathsf{A}_i(0, t, \mathcal{I})$ and $\mathsf{A}_i(0, t, \mathcal{S})$ in (2), we have $\mathsf{lag}_i(t, \mathcal{S}) = (t - \Phi_i)u_i - (d_{i,k} - \Phi_i)u_i = (t - d_{i,k})u_i$. ◀

For the task set in Ex. 3 and its GEDF schedule in Fig. 1(a), τ_1's active job at time 2 is $\tau_{1,1}$ and it has no pending job at time 2 in $\mathcal{S}$. The lag of τ_1 at time 2 in $\mathcal{S}$ is $\mathsf{lag}_1(2, \mathcal{S}) = (2 - 3)\frac{2}{3} = -\frac{2}{3}$.

▶ **Lemma 8.** *If $\tau_{i,k}$ completes execution at or before $t \geq \Phi_i$ in $\mathcal{S}$, then $\mathsf{lag}_i(t, \mathcal{S}) \leq (t - d_{i,k})u_i$.*

Proof. Since $\tau_{i,k}$ completes execution at or before t, all jobs of τ_i released at or before $r_{i,k}$ complete execution at or before t. Hence, $\mathsf{A}_i(0, t, \mathcal{S}) \geq \sum_{j=1}^{k} C_i = \sum_{j=1}^{k} T_i u_i = \sum_{j=1}^{k} (r_{i,j+1} - r_{i,j})u_i = (r_{i,k+1} - r_{i,1})u_i = (d_{i,k} - \Phi_i)u_i$. By the definition of $\mathcal{I}$, we have $\mathsf{A}_i(0, t, \mathcal{I}) = (t - \Phi_i)u_i$. Substituting $\mathsf{A}_i(0, t, \mathcal{I})$ and $\mathsf{A}_i(0, t, \mathcal{S})$ in (2), we have $\mathsf{lag}_i(t, \mathcal{S}) = \mathsf{A}_i(0, t, \mathcal{I}) - \mathsf{A}_i(0, t, \mathcal{S}) \leq (t - \Phi_i)u_i - (d_{i,k} - \Phi_i)u_i = (t - d_{i,k})u_i$. ◀

▶ **Lemma 9.** *If τ_i has a pending job $\tau_{i,k}$ at $t \geq \Phi_i$ in $\mathcal{S}$, then $\mathsf{lag}_i(t, \mathcal{S}) > (t - d_{i,k})u_i$.*

Proof. Since $\tau_{i,k}$ is pending at time t, we have $\mathsf{A}_i(0,t,\mathcal{S}) < \sum_{j=1}^{k} C_i = \sum_{j=1}^{k} T_i u_i = \sum_{j=1}^{k}(r_{i,j+1} - r_{i,j})u_i = (r_{i,k+1} - r_{i,1})u_i = (d_{i,k} - \Phi_i)u_i$. By the definition of $\mathcal{I}$, $\mathsf{A}_i(0,t,\mathcal{I}) = (t-\Phi_i)u_i$ holds. Substituting $\mathsf{A}_i(0,t,\mathcal{I})$ and $\mathsf{A}_i(0,t,\mathcal{S})$ in (2), we have $\mathsf{lag}_i(t,\mathcal{S}) = \mathsf{A}_i(0,t,\mathcal{I}) - \mathsf{A}_i(0,t,\mathcal{S}) > (t-\Phi_i)u_i - (d_{i,k}-\Phi_i)u_i = (t-d_{i,k})u_i$. ◀

▶ **Lemma 10.** *If $\tau_{i,k}$ is the earliest pending job of τ_i at $t \geq \Phi_i$ in $\mathcal{S}$, then $\mathsf{lag}_i(t,\mathcal{S}) \leq (t - r_{i,k})u_i$.*

Proof. Since $\tau_{i,k}$ is the earliest pending job of τ_i at t, all jobs of τ_i prior to $\tau_{i,k}$ complete execution at or before t. Thus, $\mathsf{A}_i(0,t,\mathcal{S}) \geq \sum_{j=1}^{k-1} C_i = \sum_{j=1}^{k-1} T_i u_i = \sum_{j=1}^{k-1}(r_{i,j+1} - r_{i,j})u_i = (r_{i,k} - r_{i,1})u_i = (r_{i,k} - \Phi_i)u_i$. By the definition of $\mathcal{I}$, $\mathsf{A}_i(0,t,\mathcal{I}) = (t-\Phi_i)u_i$ holds. Substituting $\mathsf{A}_i(0,t,\mathcal{I})$ and $\mathsf{A}_i(0,t,\mathcal{S})$ in (2), we have $\mathsf{lag}_i(t,\mathcal{S}) \leq (t-\Phi_i)u_i - (r_{i,k}-\Phi_i)u_i = (t-r_{i,k})u_i$. ◀

For the task system in Ex. 3 and its GEDF schedule in Fig. 1(a), $\tau_{3,1}$ is τ_3's earliest pending job at at time 4. τ_3's lag at time 4 is $4 \times 2/3 - 1 = 5/3$. By Lemma 9, $\mathsf{lag}_3(4,\mathcal{S}) = 5/3 > (4-6) \times 2/3 = -4/3$. By Lemma 10, $\mathsf{lag}_3(4,\mathcal{S}) = 5/3 \leq (4-0) \times 2/3 = 8/3$.

Using Lemmas 7–10, we now prove Lemmas 11–14, which pertain to the relationship between the lag of a task τ_i at a pair of time instants that are separated by an integer multiple of τ_i's period. For any integer c and any pair of time instants $t, t + cT_i \geq \Phi_i$, the active jobs of τ_i at t and $t + cT_i$ receive the same allocation in $\mathcal{I}$ by time t and $t + cT_i$, respectively. If τ_i's active job $\tau_{i,k}$ at t completes execution in $\mathcal{S}$ at or before t, then $\tau_{i,k+c}$ cannot receive more allocation by time $t + cT_i$ than $\tau_{i,k}$ receives by t. The following lemma pertains to this scenario.

▶ **Lemma 11.** *For any time t and integer c such that $\min\{t, t + cT_i\} \geq \Phi_i$, if τ_i has no pending job at t in $\mathcal{S}$, then $\mathsf{lag}_i(t,\mathcal{S}) \leq \mathsf{lag}_i(t + cT_i, \mathcal{S})$.*

Proof. Let $\tau_{i,k}$ be the active job of τ_i at t, i.e., $r_{i,k} \leq t < d_{i,k}$. Since τ_i has no pending job at $t \geq \Phi_i$, by Lemma 7 we have

$$\mathsf{lag}_i(t,\mathcal{S}) = (t - d_{i,k})u_i. \tag{6}$$

Since the jobs of a task are released periodically and $t + cT_i \geq \Phi_i$ holds, $\tau_{i,k+c}$ is the active job of τ_i at time $t + cT_i$. By Lemmas 7 and 9, we have

$$\begin{aligned}
\mathsf{lag}_i(t + cT_i, \mathcal{S}) &\geq (t + cT_i - d_{i,k+c})u_i \\
&= \{\text{Since } \tau_i \text{ releases periodically, } d_{i,k+c} = d_{i,k} + cT_i\} \\
&\quad (t + cT_i - d_{i,k} - cT_i)u_i \\
&= (t - d_{i,k})u_i \\
&= \{\text{By (6)}\} \\
&\quad \mathsf{lag}_i(t,\mathcal{S}).
\end{aligned}$$

◀

For the task system in Ex. 3 and its GEDF schedule in Fig. 1(a), τ_2 has no pending job at time 5 but has a pending job at time 8 in $\mathcal{S}$. By Lemma 11, $\mathsf{lag}_2(5,\mathcal{S}) = -2/3 \leq 4/3 = \mathsf{lag}_2(8,\mathcal{S})$.

The following lemma considers the case when $\mathsf{lag}_i(t,\mathcal{S})$ is not larger than $\mathsf{lag}_i(t + cT_i, \mathcal{S})$. Informally, for any non-negative integer c, τ_i receives no more than $cT_iu_i = c \cdot C_i$ units of allocation over the interval $[t, t + cT_i)$. Therefore, if $\tau_{i,k}$ is pending at t, then $\tau_{i,k+c}$ must also be pending at $t + cT_i$.

▶ **Lemma 12.** *For any integer c such that $\min\{t, t+cT_i\} \geq \Phi_i$, if $\mathsf{lag}_i(t,\mathcal{S}) \leq \mathsf{lag}_i(t+cT_i,\mathcal{S})$ holds and $\tau_{i,k}$ is the earliest pending job of τ_i at t in $\mathcal{S}$, then $\tau_{i,k+c}$ is pending at $t+cT_i$ in $\mathcal{S}$.*

Proof. Assume for a contradiction that $\tau_{i,k+c}$ is not pending at time $t + cT_i$. Since $\tau_{i,k}$ is pending at $t \geq \Phi_i$, $r_{i,k} \leq t$ holds, and by Lemma 9 we have

$$\mathsf{lag}_i(t, \mathcal{S}) > (t - d_{i,k})u_i. \tag{7}$$

Since jobs are released periodically and $r_{i,k} \leq t$ holds, we have $r_{i,k+c} \leq t + cT_i$. Thus, $\tau_{i,k+c}$ finishes execution at or before $t + cT_i$ (as it is not pending then). By Lemma 8, we have

$$\begin{aligned}
\mathsf{lag}_i(t + cT_i, \mathcal{S}) &\leq (t + cT_i - d_{i,k+c})u_i \\
&= \{\text{Since } \tau_i \text{ releases periodically, } d_{i,k+c} = d_{i,k} + cT_i\} \\
&\quad (t + cT_i - d_{i,k} - cT_i)u_i \\
&= (t - d_{i,k})u_i \\
&< \{\text{By (7)}\} \\
&\quad \mathsf{lag}_i(t, \mathcal{S}),
\end{aligned}$$

a contradiction. ◀

For the task system in Ex. 3 and its GEDF schedule in Fig. 1(a), $\mathsf{lag}_2(4, \mathcal{S}) = -1/3 \leq 2/3 = \mathsf{lag}_2(7, \mathcal{S})$. By Lemma 12, since $\tau_{2,2}$ is the earliest pending job of τ_2 at time 4 in $\mathcal{S}$ and time 7 corresponds to $c = 1$, $\tau_{2,3}$ is pending at time 7 in $\mathcal{S}$.

Similarly, we consider the case where $\mathsf{lag}_i(t, \mathcal{S})$ is either not smaller or larger than $\mathsf{lag}_i(t + cT_i, \mathcal{S})$.

▶ **Lemma 13.** *For any integer c such that $\min\{t, t + cT_i\} \geq \Phi_i$, if $\tau_{i,k}$ is the earliest pending job of τ_i at time t in $\mathcal{S}$, then the following hold.*
(a) If $\mathsf{lag}_i(t, \mathcal{S}) \geq \mathsf{lag}_i(t + cT_i, \mathcal{S})$, then all jobs of τ_i released before $r_{i,k+c}$ complete execution at or before $t + cT_i$ in $\mathcal{S}$.
(b) If $\mathsf{lag}_i(t, \mathcal{S}) > \mathsf{lag}_i(t + cT_i, \mathcal{S})$, then all jobs of τ_i released before $r_{i,k+c}$ complete execution before $t + cT_i$ in $\mathcal{S}$.

Proof. If $k + c = 1$, then $r_{i,k+c} = r_{i,1} = \Phi_i$ and the lemma trivially holds. So assume $k + c > 1$. Since $\tau_{i,k}$ is the earliest pending job at $t \geq \Phi_i$, by Lemma 10,

$$\mathsf{lag}_i(t, \mathcal{S}) \leq (t - r_{i,k})u_i. \tag{8}$$

(a) Assume for a contradiction that τ_i has a job that is released before $r_{i,k+c}$ but does not complete execution at or before $t + cT_i$. Therefore, $\tau_{i,k+c-1}$ does not complete execution at or before $t + cT_i$ as the jobs of each task are sequential. Since $\tau_{i,k}$ is the earliest pending job of τ_i at t, we have $r_{i,k} \leq t$. Since jobs are released periodically, we have $r_{i,k+c} \leq t + cT_i$, which implies $r_{i,k+c-1} \leq t + cT_i$. Therefore, $\tau_{i,k+c-1}$ is pending at $t + cT_i$. Thus, by Lemma 9,

$$\begin{aligned}
\mathsf{lag}_i(t + cT_i, \mathcal{S}) &> (t + cT_i - d_{i,k+c-1})u_i \\
&= \{\text{Since } \tau_i \text{ releases periodically, } d_{i,k+c-1} = d_{i,k} + (c-1)T_i\} \\
&\quad (t + cT_i - d_{i,k} - (c-1)T_i)u_i \\
&= (t - d_{i,k} + T_i)u_i \\
&= \{\text{Since } r_{i,k} = d_{i,k} - T_i\} \\
&\quad (t - r_{i,k})u_i \\
&\geq \{\text{By (8)}\} \\
&\quad \mathsf{lag}_i(t, \mathcal{S}),
\end{aligned}$$

a contradiction.
(b) Since $\mathsf{lag}_i(t, \mathcal{S}) > \mathsf{lag}_i(t+cT_i, \mathcal{S})$, by (a), all jobs of τ_i released before $r_{i,k+c}$ finish execution at or before $t + cT_i$. Assume that they do not complete execution before $t + cT_i$. Thus, they complete execution at $t + cT_i$, and no job released at or after $r_{i,k+c}$ executes at or before $t + cT_i$. Thus, $\mathsf{A}_i(0, t + cT_i, \mathcal{S}) = \sum_{j=1}^{k+c-1} C_i = \sum_{j=1}^{k+c-1} T_i u_i = \sum_{j=1}^{k+c-1} (r_{i,j+1} - r_{i,j}) u_i = (r_{i,k+c} - \Phi_i) u_i = (r_{i,k} + cT_i - \Phi_i) u_i$. Thus, by the definition of $\mathcal{I}$ and (2), $\mathsf{lag}_i(t + cT_i, \mathcal{S}) = (t + cT_i - \Phi_i) u_i - (r_{i,k} + cT_i - \Phi_i) u_i = (t - r_{i,k}) u_i \geq \mathsf{lag}_i(t, \mathcal{S})$, a contradiction. ◀

For the task system in Ex. 3 and its GEDF schedule in Fig. 1(a), $\mathsf{lag}_2(8, \mathcal{S}) = 4/3 > 1/3 = \mathsf{lag}_2(11, \mathcal{S})$. By Lemma 13(b), since $\tau_{2,3}$ is the earliest pending job of τ_2 at time 8 in $\mathcal{S}$ and time 11 corresponds to $c = 1$, all jobs of τ_2 prior to $\tau_{2,4}$ complete execution before time 11 in $\mathcal{S}$.

We now utilize Lemmas 12 and 13(a) to establish a necessary condition for $\mathsf{lag}_i(t, \mathcal{S}) = \mathsf{lag}_i(t + cT_i, \mathcal{S})$ to hold. Intuitively, if $\mathsf{lag}_i(t, \mathcal{S}) = \mathsf{lag}_i(t + cT_i, \mathcal{S})$ holds, then in $\mathcal{S}$ any job $\tau_{i,k}$'s allocation at or before t must equal the allocation of job $\tau_{i,k+c}$ at or before $t + cT_i$.

▶ **Lemma 14.** *For any time t and integer c such that $\min\{t, t + cT_i\} \geq \Phi_i$, if $\mathsf{lag}_i(t, \mathcal{S}) = \mathsf{lag}_i(t + cT_i, \mathcal{S})$, then the following hold.*
(a) If there is no pending job of τ_i at t in $\mathcal{S}$, then there is no pending job of τ_i at $t + cT_i$ in $\mathcal{S}$.
(b) If $\tau_{i,k}$ is the earliest pending job of τ_i at t in $\mathcal{S}$, then $\tau_{i,k+c}$ is the earliest pending job of τ_i at $t + cT_i$ in $\mathcal{S}$.

Proof. **(a)** Assume that there is a pending job of τ_i at $t + cT_i$ and let $\tau_{i,k}$ be the earliest pending job of τ_i at $t + cT_i$. Substituting t and c in Lemma 12 by $t + cT_i$ and $-c$, respectively, job $\tau_{i,k-c}$ is pending at t, a contradiction.
(b) By Lemma 12, $\tau_{i,k+c}$ is pending at $t + cT_i$. By Lemma 13(a), all jobs of τ_i released before $r_{i,k+c}$ finish execution at or before $t + cT_i$. Thus, $\tau_{i,k+c}$ is the earliest pending job of τ_i at $t + cT_i$. ◀

We now give a necessary condition for the lag-monotonicity property to not hold.

▶ **Lemma 15.** *Let $t \geq \Phi_i + T_{max}$ be the first time instant (if one exists) such that $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) > \mathsf{lag}_i(t, \mathcal{S})$ holds in $\mathcal{S}$. Then, the following hold.*
(a) $t > \Phi_i + T_{max}$.
(b) τ_i executes during $[t - 1, t)$, but does not execute during $[t - T_{max} - 1, t - T_{max})$ in $\mathcal{S}$.

Proof. **(a)** Assume that $t = \Phi_i + T_{max}$. Since $t - T_{max} = \Phi_i$, we have $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) = \mathsf{lag}_i(\Phi_i, \mathcal{S}) = 0$. Since T_i divides T_{max}, by Lemma 6, we have $\mathsf{lag}_i(t, \mathcal{S}) = \mathsf{lag}_i(\Phi_i + T_{max}, \mathcal{S}) \geq 0$. Therefore, $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) \leq \mathsf{lag}_i(t, \mathcal{S})$, a contradiction.
(b) By (a), $t - 1 \geq \Phi_i + T_{max}$ holds. By the definition of t, we have

$$\mathsf{lag}_i(t - T_{max} - 1, \mathcal{S}) \leq \mathsf{lag}_i(t - 1, \mathcal{S}). \tag{9}$$

Assume that τ_i does not execute during $[t-1, t)$ or does execute during $[t-T_{max}-1, t-T_{max})$. Then, one of the following three cases holds.

Case 1. *τ_i executes during both $[t - T_{max} - 1, t - T_{max})$ and $[t - 1, t)$.* By Lemma 4(a),

$$\mathsf{lag}_i(t - T_{max}, \mathcal{S}) = \mathsf{lag}_i(t - T_{max} - 1, \mathcal{S}) + u_i - 1, \tag{10}$$

and

$$\mathsf{lag}_i(t, \mathcal{S}) = \mathsf{lag}_i(t - 1, \mathcal{S}) + u_i - 1. \tag{11}$$

Since $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) > \mathsf{lag}_i(t, \mathcal{S})$, by (10) and (11), we have $\mathsf{lag}_i(t - T_{max} - 1, \mathcal{S}) > \mathsf{lag}_i(t - 1, \mathcal{S})$, which contradicts (9).

Case 2. *τ_i does not execute during both $[t-T_{max}-1, t-T_{max})$ and $[t-1, t)$.* By Lemma 4(b),

$$\mathsf{lag}_i(t - T_{max}, \mathcal{S}) = \mathsf{lag}_i(t - T_{max} - 1, \mathcal{S}) + u_i, \tag{12}$$

and

$$\mathsf{lag}_i(t, \mathcal{S}) = \mathsf{lag}_i(t - 1, \mathcal{S}) + u_i. \tag{13}$$

Since $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) > \mathsf{lag}_i(t, \mathcal{S})$, by (12) and (13), we have $\mathsf{lag}_i(t - T_{max} - 1, \mathcal{S}) > \mathsf{lag}_i(t - 1, \mathcal{S})$, which contradicts (9).

Case 3. *τ_i executes during $[t - T_{max} - 1, t - T_{max})$ but does not execute during $[t - 1, t)$.* Thus, (10) and (13) hold. Therefore, by (10), we have

$$\begin{aligned}
\mathsf{lag}_i(t - T_{max} - 1, \mathcal{S}) &= \mathsf{lag}_i(t - T_{max}, \mathcal{S}) + 1 - u_i \\
&\geq \{\text{Since } u_i \leq 1\} \\
&\quad \mathsf{lag}_i(t - T_{max}, \mathcal{S}) \\
&> \{\text{By the definition of } t\} \\
&\quad \mathsf{lag}_i(t, \mathcal{S}) \\
&\geq \{\text{By (13) and } u_i \geq 0\} \\
&\quad \mathsf{lag}_i(t - 1, \mathcal{S}),
\end{aligned}$$

a contradiction to (9). ◀

▶ **Definition 16.** *Let $h_i = T_{max}/T_i$.*

The following lemma gives a lag-monotonicity property for SRT-schedulable systems that is similar to one given previously for HRT-schedulable systems under a job-level fixed-priority scheduler [7]. Informally, we show that, using Lemmas 11–13 and 15, no task can receive more allocation in $\mathcal{S}$ than $\mathcal{I}$ over an interval $[t - T_{max}, t)$ because of the existence of higher-priority jobs of other tasks, i.e., over-allocating a task would require under-allocating another task, violating the priority ordering of the jobs.

▶ **Lemma 17.** *For any task τ_i and any time $t \geq \Phi_i + T_{max}$, $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) \leq \mathsf{lag}_i(t, \mathcal{S})$.*

Proof. We use Fig. 2 to illustrate the proof. Assume for a contradiction that t is the first time instant such that $t \geq \Phi_i + T_{max}$ and there is a task τ_i with $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) > \mathsf{lag}_i(t, \mathcal{S})$. By Lemma 15(b), τ_i executes during $[t - 1, t)$. Let $\tau_{i,p}$ be the job of τ_i that executes during $[t - 1, t)$. Since T_i divides T_{max}, by the contrapositive of Lemma 11 (with t and c replaced by $t - T_{max}$ and h_i, respectively), there is a pending job of τ_i at $t - T_{max}$. Let $\tau_{i,k}$ be the earliest pending job of τ_i at $t - T_{max}$.

▷ Claim. $r_{i,k} < t - T_{max}$.

Proof. Assume otherwise. Then, $r_{i,k} = t - T_{max}$ and $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) = 0$ hold. Since jobs are released periodically and $t = r_{i,k} + T_{max}$ holds, there is a non-negative integer c such that $t = \Phi_i + cT_i$, which by Lemma 6 implies $\mathsf{lag}_i(t, \mathcal{S}) = \mathsf{lag}_i(\Phi_i + cT_i, \mathcal{S}) \geq 0$. Therefore, $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) = 0 \leq \mathsf{lag}_i(t, \mathcal{S})$, and t cannot be a time instant with $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) > \mathsf{lag}_i(t, \mathcal{S})$. Therefore, $r_{i,k} < t - T_{max}$ holds. ◀

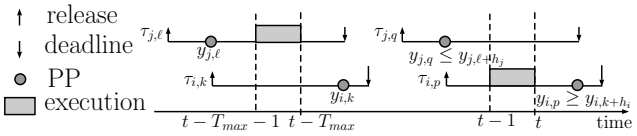


Figure 2 Illustration of the proof of Lemma 17.

By the above claim, $\tau_{i,k}$ is pending at $t - T_{max} - 1$. By Lemma 15(b), $\tau_{i,k}$ does not execute during $[t - T_{max} - 1, t - T_{max})$ (see Fig. 2). Since $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) > \mathsf{lag}_i(t, \mathcal{S})$, substituting t and c in Lemma 13(b) by $t - T_{max}$ and h_i (Def. 16), respectively, all jobs of τ_i released before $r_{i,k+h_i}$ complete execution before t (at or before $t-1$). Thus, $p \geq k + h_i$ and we have

$$\begin{aligned} y_{i,p} &\geq y_{i,k+h_i} \\ &= \{\text{Since } \tau_i \text{ releases periodically, } y_{i,k+h_i} = y_{i,k} + h_i T_i \text{ holds and by Def. 16}\} \\ &\quad y_{i,k} + T_{max}. \end{aligned} \tag{14}$$

Since $\tau_{i,k}$ is pending but does not execute during $[t - T_{max} - 1, t - T_{max})$ and $\tau_{i,p}$ executes during $[t-1, t)$, there must be a task τ_j that executes during $[t - T_{max} - 1, t - T_{max})$, but does not execute during $[t-1, t)$. Let $\tau_{j,\ell}$ executes during $[t - T_{max} - 1, t - T_{max})$ (see Fig. 2). By Lemma 15(a), $t > \Phi_i + T_{max}$, and hence, $t - 1 \geq \Phi_i + T_{max}$. Thus, by the definition of t, $\mathsf{lag}_j(t - T_{max} - 1, \mathcal{S}) \leq \mathsf{lag}_j(t-1, \mathcal{S})$ holds. Since $\tau_{j,\ell}$ executes during $[t - T_{max} - 1, t - T_{max})$, it is the earliest pending job of τ_j at $t - T_{max} - 1$. Substituting $\tau_{i,k}, t$, and c in Lemma 12 by $\tau_{j,\ell}, t - T_{max} - 1$, and h_j, respectively, $\tau_{j,\ell+h_j}$ is pending at $t - 1$. Therefore, τ_j has a pending job at $t - 1$; let $\tau_{j,q}$ be the earliest pending job of τ_j at $t - 1$. Thus, we have

$$\begin{aligned} y_{j,q} &\leq y_{j,\ell+h_j} \\ &= \{\text{Since } \tau_j \text{ releases periodically, } y_{j,\ell+h_j} = y_{j,\ell} + h_j T_j \text{ holds and by Def. 16}\} \\ &\quad y_{j,\ell} + T_{max}. \end{aligned} \tag{15}$$

Since $\tau_{i,k}$ is the earliest pending job of τ_i at $t - T_{max} - 1$ but does not execute during $[t - T_{max} - 1, t - T_{max})$, and $\tau_{j,\ell}$ executes during $[t - T_{max} - 1, t - T_{max})$, we have two cases.

Case 1. $y_{j,\ell} < y_{i,k}$. Substituting $y_{j,\ell}$ by $y_{i,k}$ in (15), we have

$$\begin{aligned} y_{j,q} &< y_{i,k} + T_{max} \\ &\leq \{\text{By (14)}\} \\ &\quad y_{i,p}. \end{aligned}$$

Therefore, $\tau_{j,q}$ has higher priority than $\tau_{i,p}$. Hence, $\tau_{i,p}$ cannot execute during $[t-1, t)$, while $\tau_{j,\ell}$ is not executing during $[t-1, t)$, a contradiction.

Case 2. $y_{j,\ell} = y_{i,k}$ and $j < i$ (as ties are broken by task index). Substituting $y_{j,\ell}$ by $y_{i,k}$ in (15), we have

$$\begin{aligned} y_{j,q} &\leq y_{i,k} + T_{max} \\ &\leq \{\text{By (14)}\} \\ &\quad y_{i,p}. \end{aligned}$$

Therefore, $\tau_{j,q}$ has higher or equal priority than $\tau_{i,p}$. Since $j < i$, $\tau_{i,p}$ cannot execute during $[t-1, t)$, while $\tau_{j,\ell}$ is not executing during $[t-1, t)$, a contradiction. ◀

By (4) and Lemma 17, we have the following LAG-monotonicity property.

▶ **Corollary 18.** *For any time instant* $t \geq \Phi_{max} + T_{max}$, $\mathsf{LAG}(t - T_{max}, \mathcal{S}) \leq \mathsf{LAG}(t, \mathcal{S})$.

For the task system in Ex. 3 and its GEDF schedule in Fig. 1(a), we have $T_{max} = 6$, $\mathsf{lag}_2(4, \mathcal{S}) = -1/3 \leq 2/3 = \mathsf{lag}_2(10, \mathcal{S})$ and $\mathsf{LAG}(4, \mathcal{S}) = 1 \leq 2 = \mathsf{LAG}(10, \mathcal{S})$.

The following lemma, proved in [28], gives a relationship between lag and the deadline of the earliest pending job of a task. The lemma, originally proved for GEDF, holds for any schedule $\mathcal{S}$ provided that tasks are periodic and (B) holds.

▶ **Lemma 19** ([28])**.** *If* $\tau_{i,k}$ *is the earliest pending job of* τ_i *at time* t *in* $\mathcal{S}$*, then*

$$d_{i,k} \leq t - \frac{\mathsf{lag}_i(t, \mathcal{S})}{u_i} + T_i. \tag{16}$$

▶ **Corollary 20.** *If* $\tau_{i,k}$ *is the earliest pending job of* τ_i *at* t *in* $\mathcal{S}$*, then* $y_{i,k} \leq t - \frac{\mathsf{lag}_i(t,\mathcal{S})}{u_i} + Y_i$.

Proof. Adding $(Y_i - T_i)$ in both side of (16), we have

$$d_{i,k} + (Y_i - T_i) \leq t - \frac{\mathsf{lag}_i(t, \mathcal{S})}{u_i} + T_i + (Y_i - T_i),$$

which by (1) and the expression $d_{i,k} = r_{i,k} + T_i$ implies

$$y_{i,k} \leq t - \frac{\mathsf{lag}_i(t, \mathcal{S})}{u_i} + Y_i. \qquad \blacktriangleleft$$

The following lemma provides a relationship between lag and tardiness. The proof of this lemma only depends on Lemma 19.

▶ **Lemma 21** ([28])**.** *If* $\mathsf{lag}_i(t, \mathcal{S}) \leq L_i$ *holds for any* t*, then the tardiness of* τ_i *is at most* $\frac{L_i}{u_i}$.

3.2 Deriving Tardiness Bounds

We now derive tardiness bounds for pseudo-harmonic periodic tasks using the properties of lag derived in Sec. 3.1. We derive our tardiness bounds by first deriving an upper bound on the lag (Lemma 27) of any task τ_i, and then applying Lemma 21 on the derived upper bound. To derive an upper bound on per-task lag, we first give Lemmas 23–26. Def. 22 is adapted from [1, 9, 20].

▶ **Definition 22.** *A time instant* t *is called* busy *if at least* $\lceil U \rceil$ *tasks have pending jobs at* t*, and* non-busy *otherwise. A time interval* $[t, t')$ *is called* busy *(resp.,* non-busy*) if each instant in the interval is busy (resp., non-busy).*

▶ **Lemma 23.** *If* τ_i *continuously executes during* $[t, t')$ *in* $\mathcal{S}$*, then* $\mathsf{lag}_i(t', \mathcal{S}) \leq \mathsf{lag}_i(t, \mathcal{S})$.

Proof. Follows from Lemma 4(a) and $u_i \leq 1$. $\blacktriangleleft$

▶ **Lemma 24.** *If* $[t, t')$ *is a busy interval in* $\mathcal{S}$*, then* $\mathsf{LAG}(t', \mathcal{S}) \leq \mathsf{LAG}(t, \mathcal{S})$.

Proof. By the definition of $\mathcal{I}$, $\mathsf{A}(t, t', \mathcal{I}) \leq U(t' - t)$ holds. By Def. 22, we have $\mathsf{A}(t, t', \mathcal{S}) \geq \lceil U \rceil (t' - t)$. Therefore, by (5) and $U \leq \lceil U \rceil$, $\mathsf{LAG}(t', \mathcal{S}) = \mathsf{LAG}(t, \mathcal{S}) + \mathsf{A}(t, t', \mathcal{I}) - \mathsf{A}(t, t', \mathcal{S}) \leq \mathsf{LAG}(t, \mathcal{S}) + U(t' - t) - \lceil U \rceil (t' - t) \leq \mathsf{LAG}(t, \mathcal{S})$. $\blacktriangleleft$

▶ **Lemma 25.** *For any* $t \geq \Phi_{max} + T_{max}$*, if* $\mathsf{LAG}(t - T_{max}, \mathcal{S}) = \mathsf{LAG}(t, \mathcal{S})$ *holds, then for each* τ_i, $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) = \mathsf{lag}_i(t, \mathcal{S})$ *holds.*

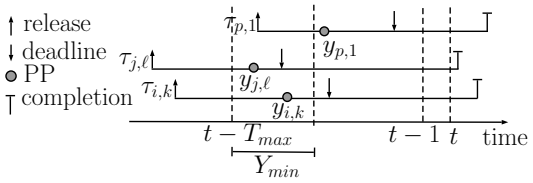


Figure 3 Illustration of the proof of Lemma 27.

Proof. Assume that there is a task τ_i with $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) \neq \mathsf{lag}_i(t, \mathcal{S})$. Since $t \geq \Phi_{max} + T_{max}$, by Lemma 17, $\mathsf{lag}_j(t - T_{max}, \mathcal{S}) \leq \mathsf{lag}_j(t, \mathcal{S})$ holds for any task τ_j including τ_i. Therefore, $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) < \mathsf{lag}_i(t, \mathcal{S})$ holds. By (4), we have

$$\begin{aligned}
\mathsf{LAG}(t - T_{max}, \mathcal{S}) &= \sum_{\tau_j \in \tau \setminus \{\tau_i\}} \mathsf{lag}_j(t - T_{max}, \mathcal{S}) + \mathsf{lag}_i(t - T_{max}, \mathcal{S}) \\
&< \{\text{Since } \mathsf{lag}_i(t - T_{max}, \mathcal{S}) < \mathsf{lag}_i(t, \mathcal{S}) \text{ and for all } j, \\
&\qquad \mathsf{lag}_j(t - T_{max}, \mathcal{S}) \leq \mathsf{lag}_j(t, \mathcal{S})\} \\
&\quad \sum_{\tau_j \in \tau \setminus \{\tau_i\}} \mathsf{lag}_j(t, \mathcal{S}) + \mathsf{lag}_i(t, \mathcal{S}) \\
&= \mathsf{LAG}(t, \mathcal{S}),
\end{aligned}$$

a contradiction. ◀

For the task system in Ex. 3 and its GEDF schedule in Fig. 1(a), $\mathsf{LAG}(7, \mathcal{S}) = 2 = \mathsf{LAG}(13, \mathcal{S})$ holds. By Lemma 25, we have $\mathsf{lag}_1(7, \mathcal{S}) = \mathsf{lag}_1(13, \mathcal{S}) = -1/3$, $\mathsf{lag}_2(7, \mathcal{S}) = \mathsf{lag}_2(13, \mathcal{S}) = 2/3$, and $\mathsf{lag}_3(7, \mathcal{S}) = \mathsf{lag}_3(13, \mathcal{S}) = 5/3$.

▶ **Lemma 26.** *For any $L_i > 0$, if t is the first time instant such that $\mathsf{lag}_i(t, \mathcal{S}) > L_i$, then τ_i does not execute during $[t - 1, t)$.*

Proof. Since $L_i > 0$ and for any $t' \leq \Phi_i$, $\mathsf{lag}_i(t', \mathcal{S}) = 0$ holds, we have $t > \Phi_i$. Therefore, $\mathsf{lag}_i(t-1, \mathcal{S}) \leq L_i$ holds. Assume that τ_i executes during $[t-1, t)$. By Lemma 23, $\mathsf{lag}_i(t, \mathcal{S}) \leq \mathsf{lag}_i(t - 1, \mathcal{S}) \leq L_i$, a contradiction. ◀

We now show that each task τ_i's lag cannot exceed $(T_{max} + Y_i - Y_{min})u_i$. Informally, assume that t is the first time instant where a task τ_i's lag exceeds $(T_{max} + Y_i - Y_{min})u_i$ in $\mathcal{S}$. If $[t-T_{max}, t)$ is a busy-interval, then by Lemma 24 (LAG does not increase over a busy interval) and Corollary 18 (LAG-monotonicity), LAG at $t - T_{max}$ and t must be the same in $\mathcal{S}$, which by Lemma 25 implies τ_i's lag at $t-T_{max}$ and t is also same. Otherwise, if there is a non-busy instant t_b in $[t - T_{max}, t)$, then by Corollary 20, τ_i's earliest pending job's priority must be higher than any job released at or after t_b throughout $[t_b, t)$. Therefore, τ_i would execute continuously throughout $[t_b, t)$, violating Lemma 26. We now give the formal proof.

▶ **Lemma 27.** *For any task τ_i and any time instant t in $\mathcal{S}$, $\mathsf{lag}_i(t, \mathcal{S}) \leq (T_{max} + Y_i - Y_{min})u_i$.*

Proof. We use Fig. 3 to illustrate the proof. Assume that there is a time instant t such that there is a task τ_i with $\mathsf{lag}_i(t, \mathcal{S}) > (T_{max} + Y_i - Y_{min})u_i$ and let t be the first such time instant. Since $\mathcal{I}$ executes τ_i at the rate of u_i, $\mathsf{lag}_i(t', \mathcal{S}) \leq T_{max}u_i$ holds for any $t' \leq \Phi_i + T_{max}$. Therefore, $t > \Phi_i + T_{max} \geq T_{max}$ holds.

We first prove that $[t - T_{max}, t)$ is a busy interval. Since $t > T_{max}$, $[t - T_{max}, t)$ is a valid time interval. By Lemma 5, there is a pending job of τ_i at t because $\mathsf{lag}_i(t, \mathcal{S}) > 0$. Let $\tau_{i,k}$

be the earliest pending job of τ_i at t. By Corollary 20, we have

$$\begin{aligned} y_{i,k} &\leq t - \frac{\mathsf{lag}_i(t,\mathcal{S})}{u_i} + Y_i \\ &< \{\text{Since } \mathsf{lag}_i(t,\mathcal{S}) > (T_{max} + Y_i - Y_{min})u_i\} \\ &\quad t - \frac{(T_{max} + Y_i - Y_{min})u_i}{u_i} + Y_i \\ &= t - T_{max} + Y_{min}. \end{aligned} \tag{17}$$

By (1), we have

$$\begin{aligned} r_{i,k} &= y_{i,k} - Y_i \\ &< \{\text{By (17)}\} \\ &\quad t - T_{max} + Y_{min} - Y_i \\ &\leq \{\text{Since } Y_{min} \leq Y_i\} \\ &\quad t - T_{max}. \end{aligned} \tag{18}$$

Thus, $\tau_{i,k}$ is pending throughout $[t - T_{max}, t)$. Since t is the first time instant with $\mathsf{lag}_i(t,\mathcal{S}) > (T_{max} + Y_i - Y_{min})u_i$, by Lemma 26, $\tau_{i,k}$ does not execute during $[t-1, t)$. Thus, there are at least m tasks with higher priority jobs than $\tau_{i,k}$ at $t-1$. Let τ^h be the set of tasks having higher priority jobs than $\tau_{i,k}$ at $t-1$. Then, $|\tau^h| \geq m$ holds. By the definition of τ^h, for any task $\tau_j \in \tau^h$, $y_{j,\ell} \leq y_{i,k}$ holds where $\tau_{j,\ell}$ is the earliest pending job of τ_j at $t-1$ (see Fig. 3). By a calculation similar to that yielding (18), $r_{j,\ell} < t - T_{max}$ holds, which implies $\tau_{j,\ell}$ is pending throughout $[t - T_{max}, t)$. Thus, by (17) we have the following property.

Property P: *Each task in $\tau^h \cup \{\tau_i\}$ has pending jobs with PPs less than $T_{max} + Y_i - Y_{min}$ throughout $[t - T_{max}, t)$.*

By Property P, $[t - T_{max}, t)$ is a busy interval. By Lemma 24, we therefore have

$$\mathsf{LAG}(t,\mathcal{S}) \leq \mathsf{LAG}(t - T_{max}, \mathcal{S}). \tag{19}$$

We now consider two cases.

Case 1. $t \geq \Phi_{max} + T_{max}$. By Corollary 18, we have

$$\mathsf{LAG}(t,\mathcal{S}) \geq \mathsf{LAG}(t - T_{max}, \mathcal{S}). \tag{20}$$

By (19) and (20), we have

$$\mathsf{LAG}(t,\mathcal{S}) = \mathsf{LAG}(t - T_{max}, \mathcal{S}). \tag{21}$$

Since $t \geq \Phi_{max} + T_{max}$ and (21) holds, by Lemma 25, $\mathsf{lag}_i(t,\mathcal{S}) = \mathsf{lag}_i(t - T_{max}, \mathcal{S})$ holds. Therefore, t cannot be the first time instant with $\mathsf{lag}_i(t,\mathcal{S}) > (T_{max} + Y_i - Y_{min})u_i$.

Case 2. $t < \Phi_{max} + T_{max}$. Let τ^s be the set of tasks such that for each $\tau_p \in \tau^s$, $t - T_{max} < \Phi_p \leq \Phi_{max}$ holds. Since each task $\tau_p \in \tau^s$ releases its first job after $t - T_{max}$, $r_{p,1} > t - T_{max}$ and $\mathsf{lag}_p(t - T_{max}, \mathcal{S}) = 0$ hold (see Fig. 3). Thus, by (1) and $Y_p \geq Y_{min}$, we have

$$(\forall \tau_p \in \tau^s : y_{p,1} > t - T_{max} + Y_{min}). \tag{22}$$

By Property P and (22), no task $\tau_p \in \tau^s$ executes during $[t - T_{max}, t)$. Therefore, we have

$$(\forall \tau_p \in \tau^s : \mathsf{lag}_p(t,\mathcal{S}) \geq 0 = \mathsf{lag}_p(t - T_{max}, \mathcal{S})). \tag{23}$$

By the definition of τ^s, for any task $\tau_q \in \tau \setminus \tau^s$, $t - T_{max} \geq \Phi_q$ holds, which implies $t \geq \Phi_q + T_{max}$. Therefore, by Lemma 17, we have

$$(\forall \tau_q \in \tau \setminus \tau^s : \mathsf{lag}_q(t, \mathcal{S}) \geq \mathsf{lag}_q(t - T_{max}, \mathcal{S})). \tag{24}$$

Since t is the first time instant with $\mathsf{lag}_i(t, \mathcal{S}) > (T_{max} + Y_i - Y_{min})u_i > 0$, $\mathsf{lag}_i(t', \mathcal{S}) \leq (T_{max} + Y_i - Y_{min})u_i$ holds for any $t' < t$. Thus, we have

$$\mathsf{lag}_i(t, \mathcal{S}) > \mathsf{lag}_i(t - T_{max}, \mathcal{S}). \tag{25}$$

By (4), we have

$$\begin{aligned}
\mathsf{LAG}(t, \mathcal{S}) &= \sum_{\tau_j \in \tau} \mathsf{lag}_j(t, \mathcal{S}) \\
&= \sum_{\tau_j \in \tau^s} \mathsf{lag}_j(t, \mathcal{S}) + \sum_{\tau_j \in \tau \setminus (\tau^s \cup \{\tau_i\})} \mathsf{lag}_j(t, \mathcal{S}) + \mathsf{lag}_i(t, \mathcal{S}) \\
&> \{\text{By (23), (24), and (25)}\} \\
&\quad \sum_{\tau_j \in \tau^s} \mathsf{lag}_j(t - T_{max}, \mathcal{S}) + \sum_{\tau_j \in \tau \setminus (\tau^s \cup \{\tau_i\})} \mathsf{lag}_j(t - T_{max}, \mathcal{S}) + \mathsf{lag}_i(t - T_{max}, \mathcal{S}) \\
&= \mathsf{LAG}(t - T_{max}, \mathcal{S}),
\end{aligned}$$

a contradiction to (19). ◀

We now give our tardiness bound in the following Theorem.

▶ **Theorem 28.** *The tardiness of task τ_i is at most $T_{max} + Y_i - Y_{min}$ in $\mathcal{S}$.*

Proof. The theorem follows from Lemmas 21 and 27. ◀

By Theorem 28, we have following tardiness bounds under GEDF and FIFO.

▶ **Theorem 29.** *The tardiness of a task τ_i in a GEDF and FIFO schedule is at most $T_{max} + T_i - T_{min}$ and T_{max}, respectively.*

Removing Assumption (B). Prior work has shown that removing Assumption (B) does not invalidate GEDF tardiness bounds because its removal cannot cause work to shift later [28]. It can be similarly removed for any GEL scheduler.

▶ **Theorem 30.** *Let τ be a periodic task set, $\mathcal{S}$ be a GEL schedule of τ satisfying (B), and $\mathcal{S}'$ be a GEL schedule with the same PP for each job of τ without satisfying (B). Then no job in $\mathcal{S}'$ finishes later than in $\mathcal{S}$.*

Tightness. The following example shows the tightness of the tardiness bound in Theorem 28.

▶ **Example 31.** Consider a task system τ with $m+1$ tasks where $\tau_i = (0, m, m+1)$. For any job-level fixed-priority scheduler, the maximum tardiness among all tasks is $m - 1 = T_{max} - 2$. For both FIFO and GEDF, the tardiness bound of τ by Theorem 28 is T_{max}. A GEDF/FIFO schedule corresponding to $m = 5$ is shown in Fig. 4. Jobs $\tau_{6,1}, \tau_{5,2}$, and $\tau_{4,3}$ have tardiness of 4.0, 3.0, and 2.0 time units, respectively. Similarly, $\tau_{3,4}$ has tardiness of 1.0 time unit (not shown in Fig. 4). τ_1 and τ_2 have no tardy job. The schedule repeats after 30.0 time units. ◀

Sporadic tasks. We can enable similar tardiness bounds for sporadic tasks using GEL-scheduled periodic servers. For each task τ_i, we create a server task $\tau_i^s = (0, C_i, T_i)$. We schedule the server tasks by a GEL scheduler where each server job of τ_i^s receives an allocation of exactly C_i time units. We schedule job $\tau_{i,k}$ on server job $\tau_{i,\ell}^s$ where $d_{i,k} \in (r_{i,\ell}^s, d_{i,\ell}^s]$ (see Fig. 5). Since both τ_i and τ_i^s have the same period, no other job of τ_i is scheduled on $\tau_{i,\ell}^s$. Since $\tau_{i,\ell}^s$ receives allocation of C_i time units, $\tau_{i,k}$ finishes execution at or before $\tau_{i,\ell}^s$ completes. Since $d_{i,\ell}^s - d_{i,k} \leq T_i$, we have the following theorem.

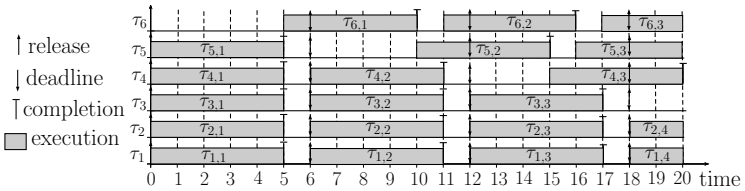


Figure 4 Schedule corresponding to Ex. 31

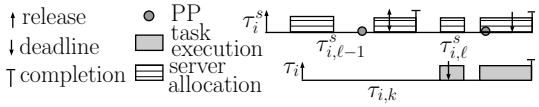


Figure 5 Scheduling sporadic tasks by GEL-scheduled periodic servers.

▶ **Theorem 32.** *A pseudo-harmonic sporadic task system τ can be scheduled using periodic servers scheduled by a GEL scheduler such that each task τ_i's tardiness is at most $T_{max} + Y_i - Y_{min} + T_i$.*

Discussion. While the tardiness bound given in Theorem 28 is tight in general, the tardiness bound is not tight for task systems with a smaller total utilization than m. For instance, a HRT-schedulable task system also has the tardiness bound specified in Theorem 28. Although the tardiness bounds in [9, 11, 27] can have smaller bounds when the total utilization is less than m compared to systems with full utilization, they also have similar issues, e.g., positive tardiness bounds for HRT-schedulable task systems.

Although the tardiness bound given in Theorem 28 is T_{max} under FIFO, the tardiness bound under GEDF can be larger than T_{max}. The tardiness of a task can actually exceed T_{max} under GEDF as illustrated in the following example.

▶ **Example 33.** Consider a task system with five tasks $\tau_1 = (1, 4, 5)$, $\tau_2 = (3, 3, 4)$, $\tau_3 = (9, 19, 25)$, $\tau_4 = (20, 99, 100)$, $\tau_5 = (75, 70, 100)$ scheduled on four processors by GEDF. It can be shown that the tardiness of the job $\tau_{4,48}$ is 104 time units, which is $T_{max} + 4$.

4 Exact Tardiness Bounds

Having derived a tardiness bound for pseudo-harmonic periodic tasks that does not depend on the processor or task count in Sec. 3, we now show how to derive an exact tardiness bound in pseudo-polynomial time. We do so by deriving an upper bound on the length of the prefix of a schedule during which tasks may experience increasing tardiness (afterwards, they do not). We show, in Lemma 39, that if there is a time instant $t \geq \Phi_{max}$ when LAG has the same values at t and $t - T_{max}$, then for any $t' \geq t$, the LAG values at t' and $t' - T_{max}$ are also equal. Intuitively, this implies that the schedule in the interval $[t - T_{max}, t)$ repeats after t. Moreover, since the lag of each task is bounded (Lemma 27), we can derive an upper bound on LAG (Lemma 38). Therefore, since LAG does not decrease over any interval of length T_{max} starting after Φ_{max} (LAG-monotonicity), there must be a finite interval $[\Phi_{max}, t')$ such that LAG strictly increases over any interval of length T_{max} in $[\Phi_{max}, t')$. We derive an upper bound on such an interval in Lemma 41. Intuitively, for each task, a job with the maximum tardiness of the task must complete at or before the schedule starts to cycle. We first consider task systems satisfying (B). We define a *max-tardiness-increasing interval* as follows.

▶ **Definition 34.** *Given a periodic task system τ, a* max-tardiness-increasing interval *in a schedule $\mathcal{S}$ is a finite interval of time $[0, t]$ such that for each task $\tau_i \in \tau$, if the maximum tardiness of τ_i's jobs that complete execution at or before t is x_i in $\mathcal{S}$, then the tardiness of τ_i is x_i in $\mathcal{S}$.*

We now derive an upper bound on the max-tardiness-increasing interval of a pseudo-harmonic periodic task system τ satisfying (B).

▶ **Definition 35.** *Let F be the sum of the $n-1$ largest values of $C_i(1-u_i)$; i.e., $F = \sum_{n-1 \text{ largest}} C_i(1-u_i)$. Let G be the sum of the $\lceil U \rceil - 1$ largest values of $(T_{max}+Y_i-Y_{min})u_i$; i.e., $G = \sum_{\lceil U \rceil - 1 \text{ largest}} (T_{max}+Y_i-Y_{min})u_i$. Let $E = \lceil F+G+1 \rceil$.*

The following lemma gives a trivial lower bound on the lag of a task at any time t in $\mathcal{S}$. A task's lag is minimum when its active job finishes execution as early as possible in $\mathcal{S}$, i.e., C_i time units after its release.

▶ **Lemma 36.** *For any task τ_i and time instant t, $\mathsf{lag}_i(t, \mathcal{S}) \geq -C_i(1-u_i)$.*

Proof. If $t \leq \Phi_i$, then $\mathsf{lag}_i(t, \mathcal{S}) = 0$, so assume $t > \Phi_i$. Let $\tau_{i,k}$ be the active job of τ_i at t and e_i be the cost of the completed portion of $\tau_{i,k}$ at or before t in $\mathcal{S}$. Therefore, $\mathsf{A}_i(0, t, \mathcal{S}) = \sum_{j=1}^{k-1} C_i + e_i$. By the definition of $\mathcal{I}$, all jobs of τ_i prior to $\tau_{i,k}$ complete execution by t in $\mathcal{I}$. Since jobs can only execute after their release, by the time $\tau_{i,k}$ executes for e_i units in $\mathcal{S}$, $\tau_{i,k}$ must execute at least $e_i u_i$ units of $\tau_{i,k}$ in $\mathcal{I}$. Therefore, $\mathsf{A}_i(0, t, \mathcal{I}) \geq \sum_{j=1}^{k-1} C_i + e_i u_i$. Substituting $\mathsf{A}_i(0, t, \mathcal{I})$ and $\mathsf{A}_i(0, t, \mathcal{S})$ in (2), we have $\mathsf{lag}_i(t, \mathcal{S}) \geq \sum_{j=1}^{k-1} C_i + e_i u_i - \sum_{j=1}^{k-1} C_i - e_i = -e_i(1-u_i)$. Since $e_i \leq C_i$, we have $\mathsf{lag}_i(t, \mathcal{S}) \geq -C_i(1-u_i)$. ◀

We now give a lower bound on LAG at Φ_{max} in $\mathcal{S}$. By the definition of Φ_{max}, there is at least one task with lag that equals 0 at Φ_{max}.

▶ **Lemma 37.** $\mathsf{LAG}(\Phi_{max}, \mathcal{S}) \geq -F$.

Proof. Let τ' be the set of tasks such that for any $\tau_i \in \tau'$, $\Phi_i = \Phi_{max}$ holds. Therefore, $\mathsf{lag}_i(\Phi_{max}, \mathcal{S}) = 0$ holds for any $\tau_i \in \tau'$. Thus, $\sum_{\tau_i \in \tau'} \mathsf{lag}_i(\Phi_{max}, \mathcal{S}) = 0$. Hence, by (4), we have $\mathsf{LAG}(\Phi_{max}, \mathcal{S}) = \sum_{\tau_i \in \tau} \mathsf{lag}_i(\Phi_{max}, \mathcal{S}) = \sum_{\tau_i \in \tau \setminus \tau'} \mathsf{lag}_i(\Phi_{max}, \mathcal{S})$, which by Lemma 36 implies, $\mathsf{LAG}(\Phi_{max}, \mathcal{S}) \geq \sum_{\tau_i \in \tau \setminus \tau'} -C_i(1-u_i)$. By the definition of Φ_{max}, $|\tau'| \geq 1$ holds. Therefore, by Def. 35, we have $\mathsf{LAG}(\Phi_{max}, \mathcal{S}) \geq -\sum_{n-1 \text{ largest}} C_i(1-u_i) = -F$. ◀

We now derive an upper bound on LAG at any time t in $\mathcal{S}$ by determining an upper bound on LAG at the latest non-busy time instant at or before t.

▶ **Lemma 38.** *For any t, $\mathsf{LAG}(t, \mathcal{S}) \leq G$.*

Proof. Let t_b be the latest non-busy time instant at or before t, otherwise let $t_b = 0$. We first derive an upper bound on $\mathsf{LAG}(t_b, \mathcal{S})$. If $t_b = 0$, then $\mathsf{LAG}(t_b, \mathcal{S}) = 0$. Otherwise, let $\tau' \subseteq \tau$ be the tasks with pending jobs at t_b. By (4),

$$
\begin{aligned}
\mathsf{LAG}(t_b, \mathcal{S}) &= \sum_{\tau_i \in \tau'} \mathsf{lag}_i(t_b, \mathcal{S}) + \sum_{\tau_i \notin \tau'} \mathsf{lag}_i(t_b, \mathcal{S}) \\
&\leq \{\text{By the contrapositive of Lemma 5, } (\forall \tau_i \notin \tau' : \mathsf{lag}_i(t_b, \mathcal{S}) \leq 0) \text{ holds}\} \\
&\quad \sum_{\tau_i \in \tau'} \mathsf{lag}_i(t_b, \mathcal{S}) \\
&\leq \{\text{By Lemma 27}\}
\end{aligned}
$$

$$
\begin{aligned}
&\sum_{\tau_i \in \tau'} (T_{max} + Y_i - Y_{min}) u_i \\
&\leq \{\text{By Def. 22, } |\tau'| < \lceil U \rceil\} \\
&\quad \sum_{\lceil U \rceil - 1 \text{ largest}} (T_{max} + Y_i - Y_{min}) u_i \\
&= \{\text{By Def. 35}\} \\
&\quad G.
\end{aligned}
$$

By Lemma 24, $\mathsf{LAG}(t, \mathcal{S}) \leq \mathsf{LAG}(t_b, \mathcal{S}) \leq G$ holds. ◀

Lemmas 37 and 38 imply that LAG cannot increase more than $F + G$ units over any interval $[\Phi_{max}, t)$. We use this fact later in Lemma 41. We now show that once $\mathsf{LAG}(t, \mathcal{S}) = \mathsf{LAG}(t - T_{max}, \mathcal{S})$ holds for some t, the equality also holds for all time instances after t. Informally, by Lemma 25, if $\mathsf{LAG}(t, \mathcal{S}) = \mathsf{LAG}(t - T_{max}, \mathcal{S})$ holds, then for any task τ_i, $\mathsf{lag}_i(t, \mathcal{S}) = \mathsf{lag}_i(t - T_{max}, \mathcal{S})$ also holds. This implies that the scheduling decisions at t are the same as at $t - T_{max}$. Therefore, the schedule in $[t - T_{max}, t)$ repeats in $[t, t + T_{max})$.

▶ **Lemma 39.** *If there is a time instant $t' \geq \Phi_{max} + T_{max}$ such that $\mathsf{LAG}(t' - T_{max}, \mathcal{S}) = \mathsf{LAG}(t', \mathcal{S})$ holds, then for any $t \geq t'$, $\mathsf{LAG}(t - T_{max}, \mathcal{S}) = \mathsf{LAG}(t, \mathcal{S})$ holds.*

Proof. Assume for a contradiction that there exists a $t \geq t'$ such that $\mathsf{LAG}(t - T_{max}, \mathcal{S}) \neq \mathsf{LAG}(t, \mathcal{S})$ and let t be the first such time instant. By the definition of t and t', $t > t' \geq \Phi_{max} + T_{max}$ and $t - 1 \geq \Phi_{max} + T_{max}$ hold. Therefore, $\mathsf{LAG}(t - T_{max} - 1, \mathcal{S}) = \mathsf{LAG}(t - 1, \mathcal{S})$ holds. Thus, by Lemma 25, we have

$$(\forall \tau_i : \mathsf{lag}_i(t - T_{max} - 1, \mathcal{S}) = \mathsf{lag}_i(t - 1, \mathcal{S})). \tag{26}$$

Since T_i divides T_{max}, by (26) and Lemma 14(a) (with t and c replaced by $t - T_{max} - 1$ and h_i, respectively), we have the following property.

Property Q: *Any task with no pending job at $t - T_{max} - 1$ has no pending job at $t - 1$.*

Let $\tau' \subseteq \tau$ be the set of tasks with pending jobs at $t - T_{max} - 1$. Let $\tau_{i,k}$ be the earliest pending job of $\tau_i \in \tau'$ at $t - T_{max} - 1$. By Def. 16, (26) and Lemma 14(b) (with t and c replaced by $t - T_{max} - 1$ and h_i, respectively), $\tau_{i,k+h_i}$ is the earliest pending job of τ_i at $t - 1$. Since τ_i releases jobs periodically, we have $y_{i,k+h_i} = y_{i,k} + h_i T_i = y_{i,k} + T_{max}$. Thus, the tasks in τ' have the same priority ordering at both $t - T_{max} - 1$ and $t - 1$, which along with Property Q implies that the same set of tasks execute during both $[t - T_{max} - 1)$ and $[t - 1, t)$. Hence, $\mathsf{A}(t - T_{max} - 1, t - T_{max}, \mathcal{S}) = \mathsf{A}(t - 1, t, \mathcal{S})$. Since $t - T_{max} - 1 \geq \Phi_{max}$, we have $\mathsf{A}(t - T_{max} - 1, t - T_{max}, \mathcal{I}) = \mathsf{A}(t - 1, t, \mathcal{I})$. Thus, by (5) we have

$$
\begin{aligned}
\mathsf{LAG}(t, \mathcal{S}) &= \mathsf{LAG}(t - 1, \mathcal{S}) + \mathsf{A}(t - 1, t, \mathcal{I}) - \mathsf{A}(t - 1, t, \mathcal{S}) \\
&= \{\text{Since } \mathsf{LAG}(t - 1, \mathcal{S}) = \mathsf{LAG}(t + T_{max} - 1)\} \\
&\quad \mathsf{LAG}(t - T_{max} - 1, \mathcal{S}) + \mathsf{A}(t - 1, t, \mathcal{I}) - \mathsf{A}(t - 1, t, \mathcal{S}) \\
&= \{\text{Since } \mathsf{A}(t - 1, t, \mathcal{S}) = \mathsf{A}(t - T_{max} - 1, t - T_{max}, \mathcal{S}) \text{ and} \\
&\qquad\qquad \mathsf{A}(t - 1, t, \mathcal{I}) = \mathsf{A}(t - T_{max} - 1, t - T_{max}, \mathcal{I})\} \\
&= \mathsf{LAG}(t - T_{max} - 1, \mathcal{S}) + \mathsf{A}(t - T_{max} - 1, t - T_{max}, \mathcal{I}) \\
&\qquad - \mathsf{A}(t - T_{max} - 1, t - T_{max}, \mathcal{S}) \\
&= \{\text{By (5)}\} \\
&\quad \mathsf{LAG}(t - T_{max}, \mathcal{S}),
\end{aligned}
$$

a contradiction. ◀

▶ **Corollary 40.** *If there is a time instant* $t' \geq \Phi_{max} + T_{max}$ *such that* $\mathsf{LAG}(t' - T_{max}, \mathcal{S}) = \mathsf{LAG}(t', \mathcal{S})$ *holds, then* $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) = \mathsf{lag}_i(t, \mathcal{S})$ *holds for any* $t \geq t'$ *and* $\tau_i \in \tau$.

Proof. Follows from Lemmas 39 and 25. ◀

For the task system in Ex. 3 and its GEDF schedule in Fig. 1(a), $\mathsf{LAG}(6, \mathcal{S}) = \mathsf{LAG}(12, \mathcal{S})$ holds. Therefore, for all task τ_i and $t \geq 12$, $\mathsf{LAG}(t - T_{max}, \mathcal{S}) = \mathsf{LAG}(t, \mathcal{S}) = 2$ and $\mathsf{lag}_i(t - T_{max}, \mathcal{S}) = \mathsf{lag}_i(t, \mathcal{S})$ hold. We now show that there is a time instant t after $\Phi_{max} + T_{max}$ and at or before $\Phi_{max} + ET_{max}$ where LAG has the same value at t and $t - T_{max}$. Therefore, the schedule starts to cycle at or before $\Phi_{max} + ET_{max}$. Intuitively, LAG must increase by at least 1.0 execution unit, if not equal, over each interval $[\Phi_{max} + iT_{max}, \Phi_{max} + (i+1)T_{max})$ where $0 \leq i < E$. Therefore, since $E = \lceil F + G + 1 \rceil$, LAG at $\Phi_{max} + ET_{max}$ must be more than G, contradicting Lemma 38.

▶ **Lemma 41.** *There is a time instant* $t \in [\Phi_{max} + T_{max}, \Phi_{max} + ET_{max}]$ *such that* $\mathsf{LAG}(t - T_{max}, \mathcal{S}) = \mathsf{LAG}(t, \mathcal{S})$ *holds.*

Proof. Assume $\mathsf{LAG}(t - T_{max}, \mathcal{S}) \neq \mathsf{LAG}(t, \mathcal{S})$ holds for all $t \in [\Phi_{max} + T_{max}, \Phi_{max} + ET_{max}]$. Let t be any arbitrary time instant in $[\Phi_{max} + T_{max}, \Phi_{max} + ET_{max}]$. Since $t \geq \Phi_{max} + T_{max}$, by Corollary 18, we have $\mathsf{LAG}(t - T_{max}, \mathcal{S}) \leq \mathsf{LAG}(t, \mathcal{S})$. Thus, $\mathsf{LAG}(t - T_{max}, \mathcal{S}) < \mathsf{LAG}(t, \mathcal{S})$ holds. Since tasks release jobs periodically and $t - T_{max} \geq \Phi_{max}$ holds, we have

$$\mathsf{A}(t - T_{max}, t, \mathcal{I}) = UT_{max}. \tag{27}$$

Since $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$ and $h_i = T_{max}/T_i$, we have $U = \frac{\sum_{i=1}^{n} h_i C_i}{T_{max}}$. Therefore, $UT_{max} = \sum_{i=1}^{n} h_i C_i$. Since both h_i and C_i are integers, UT_{max} is also an integer. By (5), we have

$$\begin{aligned}
\mathsf{A}(t - T_{max}, t, \mathcal{S}) &= \mathsf{A}(t - T_{max}, t, \mathcal{I}) + \mathsf{LAG}(t - T_{max}, \mathcal{S}) - \mathsf{LAG}(t, \mathcal{S}) \\
&< \{\text{Since } \mathsf{LAG}(t - T_{max}, \mathcal{S}) < \mathsf{LAG}(t, \mathcal{S})\} \\
&\quad \mathsf{A}(t - T_{max}, t, \mathcal{I}) \\
&= \{\text{Since } \mathsf{A}(t - T_{max}, t, \mathcal{I}) = UT_{max}\} \\
&\quad UT_{max}
\end{aligned} \tag{28}$$

Since UT_{max} and $\mathsf{A}(t - T_{max}, t, \mathcal{S})$ are integers, by (28) we have

$$\mathsf{A}(t - T_{max}, t, \mathcal{S}) \leq UT_{max} - 1. \tag{29}$$

Now, by (5), we have

$$\begin{aligned}
\mathsf{LAG}(\Phi_{max} + ET_{max}, \mathcal{S}) &= \mathsf{LAG}(\Phi_{max}, \mathcal{S}) + \mathsf{A}(\Phi_{max}, \Phi_{max} + ET_{max}, \mathcal{I}) \\
&\qquad - \mathsf{A}(\Phi_{max}, \Phi_{max} + ET_{max}, \mathcal{S}) \\
&= \{\text{Since } [\Phi_{max}, \Phi_{max} + ET_{max}) = \\
&\qquad \cup_{i=0}^{E-1} [\Phi_{max} + iT_{max}, \Phi_{max} + (i+1)T_{max}).\} \\
&\quad \mathsf{LAG}(\Phi_{max}, \mathcal{S}) + \sum_{i=0}^{E-1} (\mathsf{A}(\Phi_{max} + iT_{max}, \Phi_{max} + (i+1)T_{max}, \mathcal{I}) \\
&\qquad - \mathsf{A}(\Phi_{max} + iT_{max}, \Phi_{max} + (i+1)T_{max}, \mathcal{S})) \\
&\geq \{\text{Substituting } t = \Phi_{max} + (i+1)T_{max} \text{ in (27) and (29)}\}
\end{aligned}$$

$$\begin{aligned}
&\mathsf{LAG}(\Phi_{max}, \mathcal{S}) + \sum_{i=0}^{E-1} (UT_{max} - UT_{max} + 1) \\
&= \mathsf{LAG}(\Phi_{max}, \mathcal{S}) + \sum_{i=0}^{E-1} 1 \\
&\geq \{\text{By Lemma 37 and Def. 35}\} \\
&\quad - F + F + G + 1 \\
&> G,
\end{aligned}$$

a contradiction to Lemma 38. ◀

We now show that a job with the maximum tardiness must complete execution at or before $\Phi_{max} + ET_{max}$ by Lemma 42 and Theorem 43.

▶ **Lemma 42.** *If there is a time instant $t' \geq \Phi_{max} + T_{max}$ such that $\mathsf{LAG}(t' - T_{max}, \mathcal{S}) = \mathsf{LAG}(t', \mathcal{S})$ holds and x_i is the maximum tardiness of any of task τ_i's jobs that complete execution at or before t' in $\mathcal{S}$, then the tardiness of τ_i is x_i in $\mathcal{S}$.*

Proof. Assume that the tardiness of τ_i is more than x_i and let $\tau_{i,k}$ be the first job with tardiness exceeding x_i. Let t be the time instant when $\tau_{i,k}$ finishes execution. Then, $t > t'$ holds. Since $\mathsf{LAG}(t' - T_{max}, \mathcal{S}) = \mathsf{LAG}(t', \mathcal{S})$ and $t - 1 \geq t'$ hold, by Corollary 40, we have $\mathsf{lag}_i(t - T_{max} - 1, \mathcal{S}) = \mathsf{lag}_i(t - 1, \mathcal{S})$. Since $h_i = T_{max}/T_i$ and $\tau_{i,k}$ is pending at $t - 1$, substituting t and c in Lemma 14(b) by $t - 1$ and $-h_i$, respectively, $\tau_{i,k-h_i}$ is pending at $t - 1 - T_{max}$. Therefore, $\tau_{i,k-h_i}$ finishes execution at or after $t - T_{max}$. Hence, we have

$$\begin{aligned}
f_{i,k-h_i} - d_{i,k-h_i} &\geq t - T_{max} - d_{i,k-h_i} \\
&= \{\text{Since } \tau_i \text{ releases periodically, } d_{i,k-h_i} = d_{i,k} - h_i T_i\} \\
&\quad t - T_{max} - d_{i,k} + h_i T_i \\
&= \{\text{By the definition of } t \text{ and Def. 16}\} \\
&\quad f_{i,k} - d_{i,k}.
\end{aligned}$$

Therefore, $\max\{0, f_{i,k-h_i} - d_{i,k-h_i}\} \geq \max\{0, f_{i,k} - d_{i,k}\}$ holds and $\tau_{i,k}$'s tardiness cannot exceed $\tau_{i,k-h_i}$'s tardiness. ◀

For the task system in Ex. 3 and its GEDF schedule in Fig. 1(a), $\mathsf{LAG}(t - T_{max}, \mathcal{S}) = \mathsf{LAG}(t, \mathcal{S})$ holds for the first time at time 12. Job $\tau_{3,1}$ has the maximum tardiness in $\mathcal{S}$.

▶ **Theorem 43.** *If the maximum tardiness of a task τ_i's jobs that completes at or before $\Phi_{max} + ET_{max}$ is x_i in $\mathcal{S}$, then the tardiness of τ_i is x_i in $\mathcal{S}$.*

Proof. By Lemma 41, there is a time instant $t \in [\Phi_{max} + T_{max}, \Phi_{max} + ET_{max}]$ such that $\mathsf{LAG}(t - T_{max}, \mathcal{S}) = \mathsf{LAG}(t, \mathcal{S})$ holds. Let z_i be the maximum tardiness of τ_i's jobs that complete execution at or before t in $\mathcal{S}$. By Lemma 42, the tardiness of τ_i is z_i. Since $t \in [\Phi_{max} + T_{max}, \Phi_{max} + ET_{max}]$, by the definition of x_i, $z_i \leq x_i$ holds. Since the tardiness of τ_i in $\mathcal{S}$ is z_i, $z_i \geq x_i$ holds. Therefore, $x_i = z_i$. ◀

By Theorems 30 and 43, if the maximum tardiness of τ_i's jobs that complete at or before $\Phi_{max} + ET_{max}$ is x_i in a GEL schedule $\mathcal{S}$ satisfying (B), then τ_i's tardiness is at most x_i in a GEL schedule $\mathcal{S}'$ not satisfying (B).

Deriving tardiness. By Theorem 43, we can determine an exact tardiness bound of each task by simulating a schedule up to time $\Phi_{max} + ET_{max}$. By Def. 35, we have

$F = \sum_{n-1 \text{ largest}} C_i(1-u_i) \leq \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} T_i u_i \leq T_{max} \sum_{i=1}^{n} u_i \leq mT_{max}$. By Def. 35, $G = \sum_{\lceil U \rceil - 1 \text{ largest}} (T_{max}+Y_i-Y_{min})u_i \leq \sum_{m-1 \text{ largest}} (T_{max}+Y_{max}) = (m-1)(T_{max}+Y_{max})$ holds. Therefore, we have $E = \lceil F + G + 1 \rceil \leq \lceil mT_{max} + (m-1)(T_{max}+Y_{max}) + 1 \rceil$. Since scheduling decisions at each time instant are determined in polynomial time, simulating a schedule up to time $\Phi_{max} + ET_{max}$ takes pseudo-polynomial time. By Lemma 42, we can terminate the simulation early at time $t \geq \Phi_{max} + T_{max}$ by checking whether $\mathsf{LAG}(t, \mathcal{S}) = \mathsf{LAG}(t - T_{max}, \mathcal{S})$ holds. This would require storing the last T_{max} values of LAG at any time. We can also store one value of LAG at any time, e.g., the last time instant that is multiple of T_{max}, and check for LAG-equality T_{max} time after the last-stored instance. This would require simulating for at most T_{max} time units more than that required when T_{max} values of LAG are stored. We note that this method can be adapted for non-pseudo-harmonic systems with T_{max} and G replaced with H and a corresponding upper bound on LAG, respectively.

5 Experiments

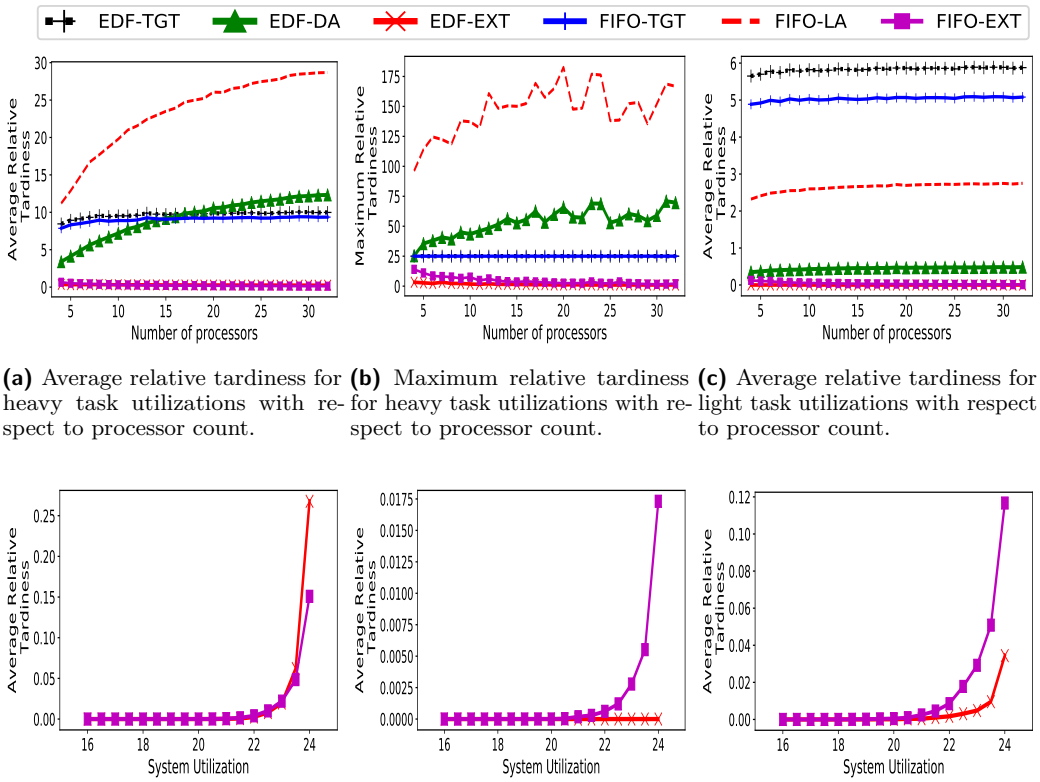
We now present the results of simulation experiments we conducted to evaluate our tardiness bounds and the effectiveness of our approach to derive exact tardiness bounds.

We generated task systems randomly for systems with 4 to 32 processors. We chose *light*, *medium*, *heavy*, or *wide* task utilizations, for which task utilizations were uniformly distributed in $[0.01, 0.3]$, $[0.3, 0.7]$, $[0.7, 1]$, and $[0.01, 1]$, respectively. We chose task periods uniformly from $\{4, 5, 10, 20, 25, 50, 100\}$ms. In case there was no task with a period of 100ms, we randomly chose a task and scaled its parameters to set its period to 100ms. We rounded down each execution cost to its nearest integer and disregarded any task if its execution cost became zero. We chose the offset of each task randomly between 0 and its period. For each utilization cap m and utilization distribution, we generated 1,000 task systems by adding tasks until five attempts to add a next task without exceeding the utilization cap failed. To compare tardiness bounds with respect to system utilization, we considered a 24-processor platform and generated 1,000 task systems for each utilization cap within $[16, 24]$ with a step size of 0.5.

We used *relative tardiness bounds* as our evaluation metric, where a task's relative tardiness is computed by dividing its tardiness by its period. For each task system, we computed exact tardiness bounds using Theorem 43 and tardiness bounds using Theorem 28 under GEDF (EDF-EXT and EDF-TGT, respectively) and FIFO (FIFO-EXT and FIFO-TGT, respectively). We also computed tardiness bounds under GEDF and FIFO using methods by Devi and Anderson [9] (EDF-DA), and Leontyev and Anderson [19] (FIFO-LA), respectively. We did not compare against the tighter bounds under GEDF from [12, 27] as they are computationally expensive to compute and have trends similar to EDF-DA [27] (In our attempt to compute tardiness bounds from [27] using the most efficient implementation from [18], we found that computing tardiness bounds for a task system on 16 or more processors can take several hours to complete). We measured the time taken to compute EDF-EXT and FIFO-EXT for each task system. We present a representative selection of our results in Fig. 6.

▶ **Observation 1.** *For heavy utilizations, the average relative tardiness bound for EDF-TGT was 7.58% smaller than for EDF-DA for large processor counts (at least 12 processors). The maximum relative tardiness bound for EDF-TGT was 56.83% smaller than for EDF-DA. The average and maximum relative tardiness bounds for FIFO-TGT were 58.47% and 83.70% smaller than for FIFO-LA, respectively.*

This can be seen in insets (a) and (b) of Fig. 6. While the mean for EDF-DA can be smaller than that for EDF-TGT for smaller processor counts (Fig. 6(a)), the maximum for EDF-DA



(a) Average relative tardiness for heavy task utilizations with respect to processor count.

(b) Maximum relative tardiness for heavy task utilizations with respect to processor count.

(c) Average relative tardiness for light task utilizations with respect to processor count.

(d) Average relative tardiness for heavy task utilizations with respect to system utilization.

(e) Average relative tardiness for light task utilizations with respect to system utilization.

(f) Average relative tardiness for wide task utilizations with respect to system utilization.

Figure 6 Average and maximum relative tardiness with respect to the number of processors and system utilizations.

is generally larger than for EDF-TGT (Fig. 6(b)). This is because EDF-DA and FIFO-LA tend to be larger when task utilizations are large.

▶ **Observation 2.** *For light utilizations, the average and maximum relative tardiness bounds for EDF-TGT were 1199% and 447% larger than for EDF-DA, respectively. The average and maximum relative tardiness bounds for FIFO-TGT were 90.47% and 13.65% larger than for FIFO-LA, respectively.*

This can be seen in Fig. 6(c). EDF-DA (resp., FIFO-LA) is tighter than EDF-TGT (resp., FIFO-TGT) for light per-task utilizations. This is because EDF-DA and FIFO-LA are functions of the sum of largest $m-1$ task utilizations, while EDF-TGT and FIFO-TGT do not depend on task utilizations. Note that it is possible to derive a tardiness bound that is a function of task utilizations by a method similar to [9,19] using the upper bound on LAG from Lemma 38.

▶ **Observation 3.** *Across all task systems, the average relative tardiness for EDF-EXT and FIFO-EXT was 0.09 and 0.17, respectively. The maximum relative tardiness for EDF-EXT and FIFO-EXT was 4.75 and 14.0, respectively. For heavy and light utilizations, the average relative tardiness for EDF-EXT was 1.11% larger and 99.9% smaller than FIFO-EXT, respectively.*

This can be seen in insets (a), (b), and (c) of Fig. 6. Average and maximum relative tardiness are usually smaller under GEDF than FIFO. However, average and maximum tardiness can be larger under GEDF than FIFO (see Ex. 33).

▶ **Observation 4.** *For heavy utilizations and high system utilization, the average relative tardiness for EDF-EXT was larger than FIFO-EXT. For the remaining utilization distributions, the average relative tardiness for EDF-EXT were smaller than FIFO-EXT.*

This can be seen in insets (d), (e), and (f) of Fig. 6. GEDF has smaller average relative tardiness than FIFO on average.

▶ **Observation 5.** *Across all task systems, the average time to compute EDF-EXT and FIFO-EXT was 386 and 64.5 ms, respectively. The maximum time to compute EDF-EXT and FIFO-EXT was 6.95 and 0.63 sec, respectively. (These computations were done on 2.50 GHz Intel processors with 30M cache and 256GB RAM.)*

This implies that exact tardiness bounds can often be efficiently computed. The running time increases when the number of processors is large. Note that the running time may increase significantly when T_{max} is large.

6 Conclusion

In this paper, we have presented a tardiness bound for pseudo-harmonic periodic tasks under GEL schedulers. This is the first tardiness bound under any practical global scheduler that does not increase with respect to the number of tasks or processors. We have shown the tightness of our bound and provided a method to determine similar tardiness bounds for pseudo-harmonic sporadic tasks. We have also provided a method to compute exact tardiness bounds for pseudo-harmonic periodic tasks under GEL schedulers.

Several other issues regarding tardiness under global schedulers remain unresolved. For example, we plan to investigate whether non-preemptive GEL schedulers provide tardiness bounds that do not depend on the processor or task count for pseudo-harmonic task systems. We also want to investigate whether a similar tardiness bound exists for non-pseudo-harmonic task systems under any GEL scheduler. Finally, we want to devise faster methods to compute exact tardiness bounds for both pseudo-harmonic and non-pseudo-harmonic task systems.

References

1. S. Ahmed and J. Anderson. A soft-real-time-optimal semi-clustered scheduler with a constant tardiness bound. In *RTCSA'20*, pages 1–10. IEEE, 2020.
2. J. Anderson and A. Srinivasan. Mixed pfair/erfair scheduling of asynchronous periodic tasks. *Journal of Computer and System Sciences*, 68(1):157–204, 2004.
3. S. Anssi, S. Kuntz, S. Gérard, and F. Terrier. On the gap between schedulability tests and an automotive task model. *Journal of Systems Architecture*, 59(6):341–350, 2013.
4. S. Baruah, N. Cohen, C. Plaxton, and D. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, 1996.
5. S. Baruah, J. Gehrke, and C. Plaxton. Fast scheduling of periodic tasks on multiple resources. In *IPSS'95*, pages 280–288. IEEE, 1995.
6. J. Busquets-Mataix, J. Serrano, R. Ors, P. Gil, and A. Wellings. Using harmonic task-sets to increase the schedulable utilization of cache-based preemptive real-time systems. In *RTCSA '96*, pages 195–202. IEEE, 1996.
7. L. Cucu-Grosjean and J. Goossens. Exact schedulability tests for real-time scheduling of periodic tasks on unrelated multiprocessor platforms. *Journal of Systems Architecture*, 57(5):561–569, 2011.
8. U. Devi and J. Anderson. Tardiness bounds under global EDF scheduling on a multiprocessor. In *RTSS '05*, pages 330–341. IEEE, 2005.

9 U. Devi and J. Anderson. Tardiness bounds under global EDF scheduling on a multiprocessor. *Real-Time Systems*, 38(2):133–189, 2008.

10 S. Dhall and C. Liu. On a real-time scheduling problem. *Operations Research*, 26(1):127–140, 1978.

11 J. Erickson, J. Anderson, and B. Ward. Fair lateness scheduling: reducing maximum lateness in G-EDF-like scheduling. *Real-Time Systems*, 50(1):5–47, 2014.

12 J. Erickson, U. Devi, and S. Baruah. Improved tardiness bounds for global EDF. In *ECRTS'10*, pages 14–23. IEEE, 2010.

13 L. Abeni et al. Deadline task scheduling. Linux kernel documentation. `https://github.com/torvalds/linux/blob/master/Documentation/scheduler/sched-deadline.rst`. [Online; accessed 06-May-2021].

14 Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. Koutsoukos, and H. Wang. Feedback thermal control for real-time systems. In *RTAS'10*, pages 111–120. IEEE, 2010.

15 J. Goossens, E. Grolleau, and L. Cucu-Grosjean. Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms. *Real-Time Systems*, 52(6):808–832, 2016.

16 C. Kenna, J. Herman, B. Brandenburg, A. Mills, and J. Anderson. Soft real-time on multiprocessors: Are analysis-based schedulers really worth it? In *RTSS'11*, pages 93–103. IEEE, 2011.

17 Simon Kramer, Dirk Ziegenbein, and Arne Hamann. Real world automotive benchmarks for free. In *WATERS'15*, 2015.

18 M. Leoncini, M. Montangero, and P. Valente. A parallel branch-and-bound algorithm to compute a tighter tardiness bound for preemptive global EDF. *Real-Time Systems*, 55(2):349–386, 2019.

19 H. Leontyev and J. Anderson. Tardiness bounds for FIFO scheduling on multiprocessors. In *ECRTS'07*, page 71. IEEE, 2007.

20 H. Leontyev and J. Anderson. Generalized tardiness bounds for global multiprocessor scheduling. *Real-Time Systems*, 44(1-3):26–71, 2010.

21 H. Li, J. Sweeney, K. Ramamritham, R. Grupen, and P. Shenoy. Real-time support for mobile robotics. In *RTAS'03*, pages 10–18. IEEE, 2003.

22 V. Nélis, P. Yomsi, and J. Goossens. Feasibility intervals for homogeneous multicores, asynchronous periodic tasks, and FJP schedulers. In *RTNS'13*, pages 277–286. ACM, 2013.

23 P. Regnier, G. Lima, E. Massa, G. Levin, and S. Brandt. RUN: optimal multiprocessor real-time scheduling via reduction to uniprocessor. In *RTSS'11*, pages 104–115. IEEE, 2011.

24 C. Shih, S. Gopalakrishnan, P. Ganti, M. Caccamo, and L. Sha. Scheduling real-time dwells using tasks with synthetic periods. In *RTSS'03*, pages 210–219. IEEE, 2003.

25 S. Tang and J. Anderson. Towards practical multiprocessor EDF with affinities. In *RTSS'20*, IEEE, pages 89–101, 2020.

26 S. Tang, S. Voronov, and J. Anderson. GEDF tardiness: Open problems involving uniform multiprocessors and affinity masks resolved. In *ECRTS'19*, pages 13:1–13:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

27 P. Valente. Using a lag-balance property to tighten tardiness bounds for global EDF. *Real-Time Systems*, 52(4):486–561, 2016.

28 K. Yang and J. Anderson. On the soft real-time optimality of global EDF on uniform multiprocessors. In *RTSS'17*, pages 319–330. IEEE, 2017.