Improved Conditions for Bounded Tardiness under EPDF Pfair Multiprocessor Scheduling☆

UmaMaheswari C. Devi[a,*,1], James H. Anderson[b]

[a] IBM India Research Laboratory,
Bangalore, KA 560071, India

[b] Department of Computer Science, The University of North Carolina,
Chapel Hill, NC 27599-3175 USA

Abstract

The earliest-pseudo-deadline-first (EPDF) Pfair algorithm is more efficient than other known Pfair scheduling algorithms, but is not optimal for scheduling recurrent real-time task systems on more than two identical processors. Although not optimal, EPDF may be preferable for real-time systems instantiated on less-powerful platforms, those with soft timing constraints, or those whose task composition can change at run-time. In prior work, Srinivasan and Anderson established a sufficient per-task utilization restriction for ensuring a tardiness of at most q quanta, where $q \geq 1$, under EPDF. They also conjectured that under this algorithm, a tardiness bound of one quantum applies to task systems that are not subject to any restriction other than the obvious restrictions, namely, that the total system utilization not exceed the available processing capacity and per-task utilizations not exceed 1.0. In this paper, we present counterexamples that show that their conjecture is false and present sufficient per-task utilization restrictions that are more liberal than theirs. For ensuring a tardiness bound of one quantum, our restriction presents an improvement of 50% over the previous one.

Key words: soft real-time, multiprocessors, Pfairness, scheduling, tardiness bounds

1. Introduction

We consider the scheduling of recurrent (*i.e.*, periodic, sporadic, or rate-based) real-time task systems on multiprocessor platforms consisting of M identical, unit-capacity processors. Pfair scheduling, originally introduced by Baruah *et al.* [4], is the only known way of *optimally* scheduling such multiprocessor task systems. (A real-time scheduling algorithm is said to be *optimal* iff it can schedule without deadline misses every task system for which some correct schedule exists.) To ensure optimality, Pfair scheduling imposes a stronger constraint on the timeliness of tasks than that mandated by periodic scheduling. Specifically, under Pfair scheduling, each task must execute at an approximately uniform rate at all times, while respecting a fixed-size allocation quantum. A task's execution rate is defined by its *weight* (*i.e.*, *utilization*). Uniform rates are ensured by subdividing each task into quantum-length *subtasks* that are subject to intermediate

☆Work supported by NSF grants CCR 9988327, ITR 0082866, CCR 0204312, CNS 0309825, and CNS 0615197, ARO grant W911NF-06-1-0425, and a grant from Intel Corporation. This work was published in preliminary form at the 12th International Workshop on Parallel and Distributed Systems [5].

*Corresponding author.

[1] The author was also supported by an IBM Ph.D. fellowship and this work was done when the author was with the University of North Carolina, Chapel Hill.

deadlines, called *pseudo-deadlines*, computed based on the task's weight. Under most known Pfair algorithms, subtasks are scheduled on an earliest-pseudo-deadline-first basis. However, to avoid deadline misses, ties among subtasks with the same deadline must be resolved carefully. In fact, tie-breaking rules are of crucial importance when devising optimal Pfair scheduling algorithms.

Motivation. The overheads associated with the tie-breaking rules of the optimal algorithms may be problematic for some applications. The *earliest-pseudo-deadline-first* (EPDF) algorithm is computationally more efficient than optimal Pfair algorithms in that it does not use any tie-breaking rule to resolve ties among subtasks with the same pseudo-deadline, but disambiguates them arbitrarily. PD^2, the most efficient of the known optimal Pfair algorithms, requires two tie-break parameters. Though these tie-break parameters can be computed for each subtask in constant time, there exist some soft and/or dynamic real-time systems in which not using any tie-breaking rule may still be preferable. Eliminating tie-breaking rules may also be preferable in embedded systems with slower processors or limited memory bandwidth.

The viability of EPDF for scheduling soft and/or dynamic real-time systems was first considered by Srinivasan and Anderson in [11], where they provided examples of such applications for which EPDF may be preferable to PD^2. Some web-hosting systems, server farms, packet processing in programmable multiprocessor-based routers, and packet transmission on multiple, parallel outgoing router links are among the examples provided by them. In these systems, *fair* resource allocation is needed, so that quality-of-service guarantees can be provided. However, an extreme notion of fairness that precludes all deadline misses is not required. Moreover, in systems such as routing networks, the inclusion of tie-breaking information in subtask priorities may result in unacceptably high space overhead.

The applications mentioned above may also be dynamic in that the set of tasks and the utilizations of tasks requiring service may change. In [11], Srinivasan and Anderson also noted that the use of tie-breaking rules may be problematic for such dynamic task systems. As they explained, it is possible to *reweight* each task whenever its utilization changes such that its next subtask deadline is preserved. If no tie-breaking information is maintained, such an approach entails very little computational overhead. However, utilization changes can cause tie-breaking information to change, so if tie-breaking rules are used, then reweighting may necessitate an $\mathcal{O}(N)$ cost for N tasks, due to the need to re-sort the scheduler's priority queue. This cost may be prohibitive if reallocations are frequent.

Motivated by the above reasons, Srinivasan and Anderson studied EPDF and they succeeded in showing that this algorithm can guarantee a tardiness (*i.e.*, lateness) bound of $q \geq 1$ quanta for every subtask, provided a certain condition holds. Their condition can be ensured by limiting each task's weight to at most $\frac{q}{q+1}$. Unfortunately, Srinivasan and Anderson left open the question of whether such weight restrictions are necessary to guarantee small bounded tardiness. Moreover, they conjectured that EPDF can ensure a tardiness bound of one quantum as long as the weight of each task does not exceed 1.0 (*i.e.*, the capacity of a single processor), and the total system utilization does not exceed the total available processing capacity.

Contributions. Our contributions in this paper are two-fold. First, we provide counterexamples that show that the above conjecture is false, and that, in general, restrictions on individual task utilizations are *necessary* to guarantee bounded tardiness under EPDF. Our second contribution is to show that a more liberal per-task weight restriction of $\frac{q+2}{q+3}$ is sufficient to ensure a tardiness of q quanta. When $q = 1$, this presents an improvement of 50% over the previous result.

The rest of the paper is organized as follows. Section 2 provides an overview of Pfair scheduling. In Section 3, the results described above are established. Section 4 concludes.

2. Background on Pfair Scheduling

This section describes some basic concepts of Pfair scheduling, provides needed background, and summarizes results from prior work reported in [1, 2, 3, 4, 10, 11]. Pfair scheduling [4, 10] can be used to schedule a *periodic*, *sporadic*, *intra-sporadic* (IS), or *generalized-intra-sporadic* (GIS) (see below) task system τ on $M \geq 1$ identical processors, each of whose processing capacity is taken to be 1.0. As explained later, in this paper, we assume that $M \geq 3$ holds. Each task T of τ is assigned a rational *weight* $wt(T) \in (0, 1]$ that denotes the fraction of a single processor it requires. In this paper, for simplicity and to avoid some boundary cases, we assume that $wt(T) < 1$ holds. The sum of the weights of all the tasks in τ, *i.e.*, the total system utilization of τ, is assumed to be at most M, which is the total available processing capacity. For a periodic or a sporadic task T, $wt(T) = T.e/T.p$, where $T.e$ and $T.p$ are the *execution cost* and inter-arrival time or *period*, respectively, of T. When scheduled under Pfair algorithms, it is required that $T.e$ and $T.p$ be specified as integers, which are interpreted as integral numbers of quanta.

A periodic or sporadic task T may be invoked zero or more times; T is periodic if any two consecutive invocations of T are separated by exactly $T.p$ time units and is sporadic if $T.p$ is a lower-bound on the inter-arrival separation. Each invocation of T is referred to as a *job* of T. The first job may be invoked or *released* at any time at or after time zero. Every job of T executes for $T.e$ time units and should complete execution within $T.p$ time units of its release, *i.e.*, every job of T has a *relative deadline* of $T.p$ time units. (In this paper, for ease of description, we assume that each job executes for exactly $T.e$ time units.) Each task is sequential, and at any time may execute on at most one processor. A task is *light* if its weight is less than 1/2, and *heavy*, otherwise.

Pfair algorithms allocate processor time in discrete quanta; the t^{th} quantum, where $t \geq 0$, spans the time interval $[t, t+1)$, and is also referred to as *slot* t. (Hence, time t refers to the beginning of slot t.) Quanta are assumed to be aligned on all processors. All references to time are non-negative integers. The interval $[t_1, t_2)$, consists of slots $t_1, t_1 + 1, \ldots, t_2 - 1$. A task may be allocated time on different processors, but not in the same slot (*i.e.*, interprocessor migration is allowed but parallelism is not). Similarly, on each processor, at most one task may be allocated in each slot. The sequence of allocation decisions over time defines a *schedule* $\mathcal{S}$. Formally, $\mathcal{S} : \tau \times \mathbf{N} \mapsto \{0, 1\}$, where $\mathbf{N}$ is the set of nonnegative integers. $\mathcal{S}(T, t) = 1$ iff T is scheduled in slot t. On M processors, $\sum_{T \in \tau} \mathcal{S}(T, t) \leq M$ holds for all t.

Periodic, sporadic, and IS task models. In Pfair scheduling, each quantum of execution of each task is referred to as a *subtask*. Each task gives rise to a potentially infinite sequence of subtasks. The i^{th} subtask of T is denoted T_i, where $i \geq 1$. If T is periodic or sporadic, then the k^{th} job of T consists of subtasks $T_{(k-1)\cdot e+1}, \ldots, T_{k\cdot e}$, where $e = T.e$ and $k \geq 1$.

Each subtask T_i has an associated *pseudo-release* $\mathsf{r}(T_i)$ and *pseudo-deadline* $\mathsf{d}(T_i)$, defined as follows. (The prefix "pseudo-" is often omitted for conciseness.)

$$\mathsf{r}(T_i) = \Theta(T_i) + \left\lfloor \frac{i-1}{wt(T)} \right\rfloor \qquad \wedge \qquad \mathsf{d}(T_i) = \Theta(T_i) + \left\lceil \frac{i}{wt(T)} \right\rceil \tag{1}$$

In the above formulas, $\Theta(T_i) \geq 0$ denotes the *offset* of T_i and is used in modeling the late releases of sporadic and IS tasks. It is well known that the sporadic model generalizes the periodic model by allowing jobs to be released "late;" the *intra-sporadic model* (IS model), proposed by Srinivasan and Anderson in [2, 10], is a further generalization that allows subtasks to be released late. The offsets of T's various subtasks are nonnegative and satisfy the following:

$$k > i \Rightarrow \Theta(T_k) \geq \Theta(T_i). \tag{2}$$

T is *periodic* if $\Theta(T_i) = c$ holds for all i (and is *synchronous* also if $c = 0$), and is *IS*, otherwise. For a sporadic task, all subtasks that belong to the same job will have equal offsets. Examples are given in insets (a) and (b) of Figure 1. Informally, the restriction in (2) on offsets implies that the separation between any pair of subtask releases is at least the separation between those releases if the task were periodic.

The interval $[\mathsf{r}(T_i), \mathsf{d}(T_i))$ is termed the *PF-window* of T_i and is denoted $\omega(T_i)$. The following lemma, concerning PF-window lengths, follows from (1).

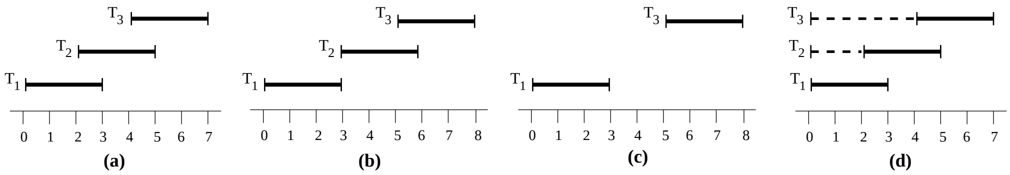

Figure 1: **(a)** PF-windows of the first job of a periodic (or sporadic) task T with weight 3/7. This job consists of subtasks T_1, T_2, and T_3, each of which must be scheduled within its window. (This pattern repeats for every job.) **(b)** PF-windows of an IS task. Subtask T_2 is released one time unit late. Here, $\Theta(T_1) = 0$ while $\Theta(T_2) = \Theta(T_3) = 1$. **(c)** PF-windows of a GIS task. Subtask T_2 is absent and subtask T_3 is released one time unit late. **(d)** PF- and IS-windows of the first job of a GIS task with early releases. All the subtasks of this job are eligible when the job arrives. (The deadline-based priority definition of the Pfair scheduling algorithms and the prohibition of parallel execution of a task ensure that the subtasks execute in the correct sequence.) For each subtask, its PF-window consists of the solid part; the IS-window includes the dashed part, in addition. For example, T_2's PF-window is $[2, 5)$ and its IS-window is $[0, 5)$.

Lemma 1. (*Anderson and Srinivasan* [3]) *The length of the PF-window of any subtask* T_i *of a task* T, $|\omega(T_i)| = \mathsf{d}(T_i) - \mathsf{r}(T_i)$, *is either* $\left\lceil \frac{1}{wt(T)} \right\rceil$ *or* $\left\lceil \frac{1}{wt(T)} \right\rceil + 1$.

GIS task model. When proving properties concerning Pfair scheduling algorithms, it is sometimes useful to "eliminate" or "omit" certain subtasks. For example, if a deadline miss does not depend on the existence of a subtask, then ignoring such a subtask makes analysis easier. In [10], Srinivasan and Anderson introduced the *generalized intra-sporadic model* (GIS model) to facilitate such selective removal of subtasks. A GIS task system is obtained by omitting subtasks from a corresponding IS (or GIS) task system. However, the spacing between subtasks of a task that are not omitted may not be decreased in comparison to how they are spaced in a periodic task. Specifically, subtask T_i may be followed by subtask T_k, where $k > i + 1$ if the following holds: $\mathsf{r}(T_k) - \mathsf{r}(T_i)$ is at least $\left\lfloor \frac{k-1}{wt(T)} \right\rfloor - \left\lfloor \frac{i-1}{wt(T)} \right\rfloor$. That is, $\mathsf{r}(T_k)$ is not smaller than what it would have been if T_{i+1}, T_{i+2}, ...,T_{k-1} were present, and released as early as possible. For the special case where T_k is the first subtask released by T, $\mathsf{r}(T_k)$ must be at least $\left\lfloor \frac{k-1}{wt(T)} \right\rfloor$. Figure 1(c) shows an example. In this example, though subtask T_2 is omitted, T_3 cannot be released before time 4. If a task T, releases subtask T_k after executing subtask T_i, then T_k is called the *successor* of T_i and T_i is called the *predecessor* of T_k. Note that a periodic task system is an IS task system, which in turn is a GIS task system, so any property established for the GIS task model applies to the other models, as well.

The early-release task model. The task models described so far are non-work-conserving in that, the second and later subtasks remain ineligible to be scheduled before their release times, even if they are otherwise ready and some processor is idle. The *early-release* (ER) task model is a work-conserving variant of the other models that allows subtasks to be scheduled before their release times [1]. Early releasing can be applied to subtasks in any of the task models considered so far, and unless otherwise specified, it should be assumed that early releasing is enabled. However, whether subtasks are actually released early is optional. To facilitate this, in this model, each subtask T_i has an eligibility time $\mathsf{e}(T_i)$ that specifies the first time slot in which T_i may be scheduled. The interval $[\mathsf{e}(T_i), \mathsf{d}(T_i))$ is referred to as the *IS-window* of T_i. Figure 1(d) gives an example. It is required that the following hold:

$$(\forall i \geq 1 :: \mathsf{e}(T_i) \leq \mathsf{r}(T_i) \ \wedge\ \mathsf{e}(T_i) \leq \mathsf{e}(T_{i+1})). \tag{3}$$

Note that the model is very flexible in that it does not preclude a job from becoming eligible before its release time, but provides mechanisms to restrict such behavior, if so desired. Such flexibility, in conjunction with the sporadic or the IS task model, can be used to schedule rate-based tasks, whose arrival pattern may be jittered, and whose instantaneous workload may deviate from the average or expected workload, such as in distributed multimedia and digital signal processing applications [7, 10].

***b*-bit.** The *b-bit* or *boundary bit* is associated with each subtask T_i and is denoted $b(T_i)$. $b(T_i)$ is as defined by (4) below.

$$b(T_i) = \left\lceil \frac{i}{wt(T)} \right\rceil - \left\lfloor \frac{i}{wt(T)} \right\rfloor. \tag{4}$$

From (1), it can be verified that if $\Theta(T_i) < \Theta(T_{i+1})$, then $\mathsf{d}(T_i) \leq \mathsf{r}(T_{i+1})$. Therefore, the PF-windows of T_i and T_{i+1} can overlap only if $\Theta(T_i) = \Theta(T_{i+1})$. It can also be verified that if $\Theta(T_i) = \Theta(T_{i+1})$, then $\mathsf{d}(T_i) - \mathsf{r}(T_{i+1}) = b(T_i)$. Hence, $b(T_i)$ determines whether the PF-window of T_i can overlap that of T_{i+1}. Observe that $b(T_i)$ is either zero or one. Therefore, the PF-windows of T_i and T_{i+1} are either disjoint or overlap by at most one slot. In Figure 1, $b(T_2) = 1$, while $b(T_3) = 0$. Therefore, the PF-window of T_2 overlaps T_3's when $\Theta(T_3) = \Theta(T_2)$ as in insets (a), (b), and (d). Further, as shown in [6], the following lemma holds.

Lemma 2. (*from* [6]) *For all* $i \geq 1$, $k \geq 1$, *the following holds.*

$$\mathsf{r}(T_{i+k}) \geq \begin{cases} \mathsf{d}(T_i) + k - 1, & b(T_i) = 0 \\ \mathsf{d}(T_i) + k - 2, & b(T_i) = 1 \end{cases}$$

Group deadlines. Like the b-bit, the group deadline is a parameter that is associated with each subtask and is used by some Pfair scheduling algorithms. The group deadline of subtask T_i is denoted $\mathsf{D}(T_i)$.

By Lemma 1, all the windows of a heavy task with weights in the range $[1/2, 1)$ are of length two or three. Informally, for such tasks, the group deadline marks the end of a sequence of subtasks whose PF-windows satisfy the following two properties: each window, except possibly of the first subtask in the sequence, is of length two, and every consecutive pair of windows is overlapping. In Figure 2(a), T_1, T_2 is one such sequence in which the first window is of length two; $T_3, \ldots, T_5$ and $T_6, \ldots, T_8$ are other such sequences in which the first window is of length three. In each sequence, if any subtask is not scheduled until its last slot, then all subsequent subtasks will be forced to be scheduled in their last slots as well, and in that sense constitute a "group." In addition, if the last subtask in the group is followed by a subtask with a window of length three, as in the first two groups considered above, then this subtask will also be precluded from being scheduled in its first slot (when any subtask in the group is scheduled in its last slot). However, no later subtask is directly impacted. Thus, the group deadline of T_i can be thought of as the earliest time t after $\mathsf{r}(T_i)$ such that t is the release time of some subtask and no subtask released at or after t is directly influenced by whether T_i is scheduled in its last slot.

Informally, for a heavy periodic task with weight less than one, the end of each slot that is not the first slot of the PF-window of any of its subtasks is a group deadline. In Figure 2(a), times 4, 8, and 11 are group deadlines for T in the interval $[0, 11]$. Note that no subtask is released at time 3, 7, or 10. Formally, the group deadline of a subtask T_i is defined as $\mathsf{D}(T_i) = (\min u : u \geq d(T_i) \wedge u \text{ is a group deadline of } T)$. For example, in Figure 2(a), $\mathsf{D}(T_1) = 4$ and $\mathsf{D}(T_6) = 11$.

The group deadline of a subtask T_i of an IS or GIS task is computed assuming that all later subtasks are present and released as early as possible, that is, under the assumption that $\Theta(T_i) = \Theta(T_j)$ holds for all $j \geq i$, regardless of how the subtasks are actually released. An illustration for an IS task is provided in Figure 2(b).

Concrete and non-concrete task systems. A task system is said to be *concrete* if release and eligibility times are specified for each subtask of each task, and *non-concrete*, otherwise. Note that an infinite number of concrete task systems can be specified for every non-concrete task system. The type of the task system is indicated only when necessary.

Pfair and ERfair schedules. The notion of a Pfair schedule is obtained by comparing the allocations that each task receives in such a schedule to those received in an ideal fluid schedule. In an ideal fluid schedule, each task executes at a precise rate given by its utilization whenever it is active. Let $\mathsf{A}(\mathsf{ideal}, T, t_1, t_2)$ and

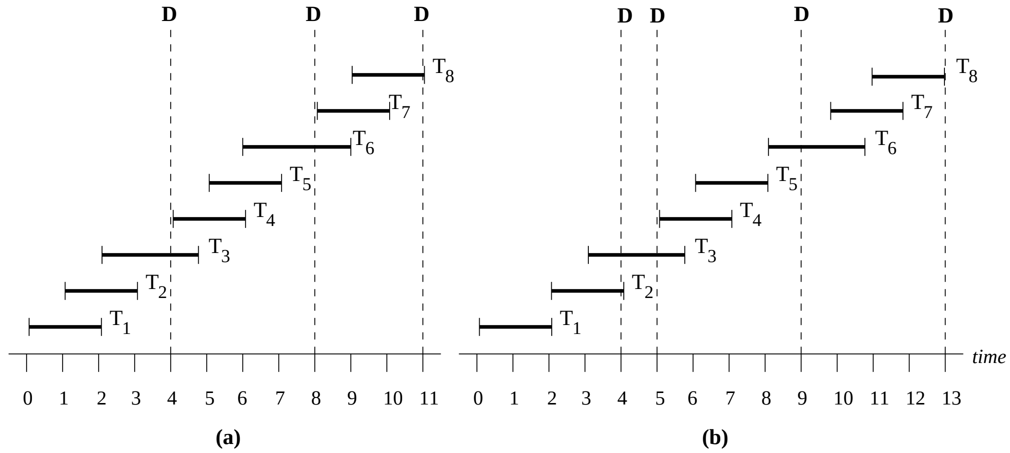

Figure 2: Illustration of group deadlines using a task T with weight $8/11$. Group deadlines are marked with a "D." **(a)** T is synchronous, periodic. The group deadlines of T_1 and T_2 are at time 4, and those of $T_3, \ldots, T_5$ and $T_6, \ldots, T_8$ are at times 8 and 11, respectively. **(b)** T is an IS task. In this example, T_2 and T_6 are released late. Nevertheless, the group deadline of T_1 is still at time 4. However, the group deadline of T_2 is at time 5. Similarly, though T_6 is released one time unit late, the group deadlines of $T_3, \ldots, T_5$ are computed under the assumption that T_6 would be released in time, and hence, are at time 9. The group deadlines of $T_6, \ldots, T_8$ are at time 13.

$\mathsf{A}(\mathcal{S}, T, t_1, t_2)$, denote the total allocation to T in the interval $[t_1, t_2)$ in the ideal schedule and an actual schedule, $\mathcal{S}$, respectively. Then, the "error" in allocation to T in $\mathcal{S}$ at time t with respect to the ideal schedule, referred to as the lag of T at t in $\mathcal{S}$, is given by $\mathsf{lag}(T, t, \mathcal{S}) = \mathsf{A}(\mathsf{ideal}, T, 0, t) - \mathsf{A}(\mathcal{S}, T, 0, t)$.

$\mathcal{S}$ is said to be a *Pfair* schedule for τ iff the following holds: $(\forall t, T \in \tau :: -1 < \mathsf{lag}(T, t, \mathcal{S}) < 1)$. Informally, each task's allocation error must always be less than one quantum. If early releases are allowed, then it is not required that the negative lag constraint, $\mathsf{lag}(T, t) > -1$, hold. A schedule for which $(\forall T, t : \mathsf{lag}(T, t) < 1)$ holds is said to be *ERfair*. The release times and deadlines in (1) are assigned such that scheduling each subtask by its deadline is sufficient to generate an ERfair schedule for τ; a Pfair schedule can be generated if each subtask is scheduled at or after its release time, as well. Further, ensuring that each task is scheduled in a Pfair or an ERfair manner is sufficient to ensure that the deadlines of all jobs are met in a periodic or sporadic task system. A schedule that is Pfair or ERfair exists for a GIS task system τ on M processors iff

$$\sum_{T \in \tau} wt(T) \leq M \tag{5}$$

holds [4, 2]. A GIS task system satisfying (5) and in which the weight of each task is at most 1.0 is said to be *feasible* on M processors. (In general, a task system is said to be *feasible* on M processors if there exists some way of correctly scheduling that task system on M processors.)

If T is synchronous and periodic, then $\mathsf{A}(\mathsf{ideal}, T, 0, t)$ equals $t \cdot wt(T)$. However, if T is GIS, then the allocation it receives in the ideal schedule may be less due to IS separations or omitted subtasks. To facilitate expressing $\mathsf{A}(\mathsf{ideal}, T, 0, t)$ for GIS tasks, let $\mathsf{A}(\mathsf{ideal}, T_i, 0, t)$ and $\mathsf{A}(\mathsf{ideal}, T_i, t)$ denote the ideal allocations to subtask T_i in $[0, t)$ and slot t, respectively. In [2], Anderson and Srinivasan showed that $\mathsf{A}(\mathsf{ideal}, T_i, t)$ is given by (6) below. An example is given in Figure 3.

$$\mathsf{A}(\mathsf{ideal}, T_i, u) = \begin{cases} \left(\left\lfloor \frac{i-1}{wt(T)} \right\rfloor + 1\right) \times wt(T) - (i-1), & u = \mathsf{r}(T_i) \\ i - \left(\left\lceil \frac{i}{wt(T)} \right\rceil - 1\right) \times wt(T), & u = \mathsf{d}(T_i) - 1 \\ wt(T), & \mathsf{r}(T_i) < u < \mathsf{d}(T_i) - 1 \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

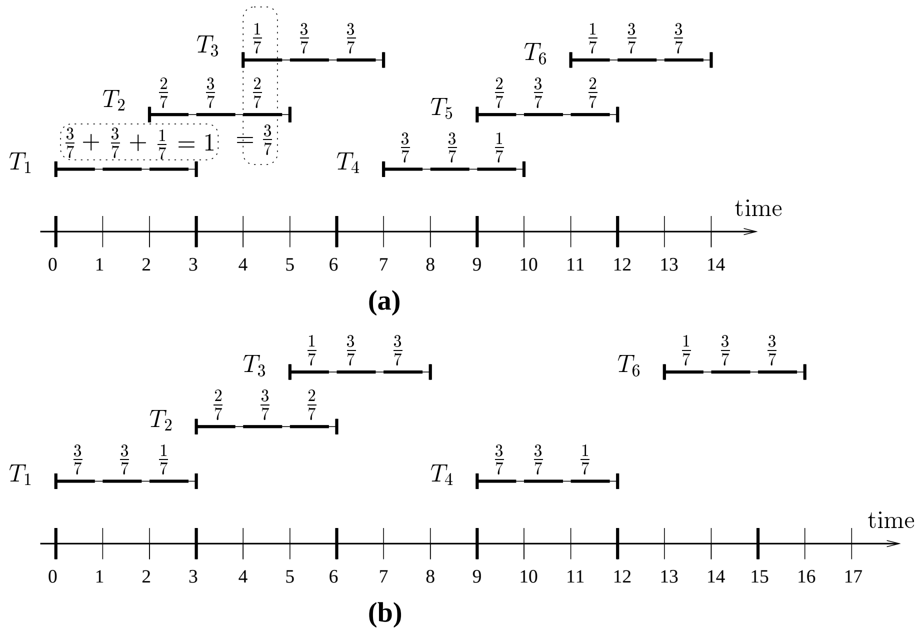

Figure 3: Per-slot ideal allocations to subtasks of a task T with weight $3/7$. These allocations are marked above the subtask windows. **(a)** T is synchronous, periodic. $\mathsf{A}(\mathsf{ideal}, T, t) = 3/7$ holds for every t. $\mathsf{A}(\mathsf{ideal}, T_2, 4) = \frac{2}{7}$ and $\mathsf{A}(\mathsf{ideal}, T_3, 4) = \frac{1}{7}$. **(b)** T is GIS. T_2's release is delayed by one time slot. T_4 is delayed by an additional time slot and T_5 is omitted. Here, $\mathsf{A}(\mathsf{ideal}, T_2, 4) = \frac{3}{7}$ and $\mathsf{A}(\mathsf{ideal}, T_3, 4) = 0$.

Let $\mathsf{A}(\mathsf{ideal}, T, t)$ denote the total allocation to task T in slot t. Then, $\mathsf{A}(\mathsf{ideal}, T, t)$ is given by $\sum_i \mathsf{A}(\mathsf{ideal}, T_i, t)$. For example, in Figure 3, $\mathsf{A}(\mathsf{ideal}, T, 4) = \mathsf{A}(\mathsf{ideal}, T_2, 4) + \mathsf{A}(\mathsf{ideal}, T_3, 4) = 1/7 + 2/7 = 3/7$, since no subtasks other than T_2 and T_3 receive a non-zero allocation in slot 4. Note that in the ideal schedule, each subtask completes executing by its deadline.

As shown in Figure 3, $\mathsf{A}(\mathsf{ideal}, T, u)$ usually equals $wt(T)$, but in certain slots, it may be less than $wt(T)$ due to omitted or delayed subtasks. Also, the total allocation that a subtask T_i receives in the slots that span its window is exactly one in the ideal schedule. These and similar properties have been proved formally in [9]. Later in this paper, we will use Lemma 3, and (7)–(10) given below (examples of which can be seen in Figure 3).

$$(\forall T, u \geq 0 :: \mathsf{A}(\mathsf{ideal}, T, u) \leq wt(T)) \tag{7}$$

$$(\forall T_i :: \sum_{u=\mathsf{r}(T_i)}^{\mathsf{d}(T_i)-1} \mathsf{A}(\mathsf{ideal}, T_i, u) = 1) \tag{8}$$

$$(\forall T_i, u \geq 0 :: \mathsf{A}(\mathsf{ideal}, T_i, u) \leq wt(T)) \tag{9}$$

$$(\forall T_i, u \in [r(T_i), d(T_i)) :: \mathsf{A}(\mathsf{ideal}, T_i, u) \geq 1/T.p) \tag{10}$$

Lemma 3. *If* $\mathsf{b}(T_i) = 1$ *and subtask* T_{i+1} *exists, then* $\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{d}(T_i) - 1) + \mathsf{A}(\mathsf{ideal}, T_{i+1}, \mathsf{r}(T_{i+1})) = wt(T)$.

A task T's ideal allocation up to time t is simply

$$\mathsf{A}(\mathsf{ideal}, T, 0, t) = \sum_{u=0}^{t-1} \mathsf{A}(\mathsf{ideal}, T, u) = \sum_{u=0}^{t-1} \sum_i \mathsf{A}(\mathsf{ideal}, T_i, u),$$

and hence

$$\begin{aligned}\mathsf{lag}(T,t,\mathcal{S}) &= \mathsf{A}(\mathsf{ideal},T,0,t) - \mathsf{A}(\mathcal{S},T,0,t) & (11)\\ &= \sum_{u=0}^{t-1}\mathsf{A}(\mathsf{ideal},T,u) - \sum_{u=0}^{t-1}\mathcal{S}(T,u) & (12)\\ &= \sum_{u=0}^{t-1}\sum_{i}\mathsf{A}(\mathsf{ideal},T_i,u) - \sum_{u=0}^{t-1}\mathcal{S}(T,u). & (13)\end{aligned}$$

From (12), $\mathsf{lag}(T,t+1)$ (the schedule parameter is omitted in the lag and LAG functions when unambiguous) is given by

$$\begin{aligned}\mathsf{lag}(T,t+1) &= \textstyle\sum_{u=0}^{t}(\mathsf{A}(\mathsf{ideal},T,u) - \mathcal{S}(T,u))\\ &= \mathsf{lag}(T,t) + \mathsf{A}(\mathsf{ideal},T,t) - \mathcal{S}(T,t). & (14)\end{aligned}$$

Similarly, by (12) again, for any $0 \le t' \le t$,

$$\begin{aligned}\mathsf{lag}(T,t+1) &= \mathsf{lag}(T,t') + \textstyle\sum_{u=t'}^{t}(\mathsf{A}(\mathsf{ideal},T,u) - \mathcal{S}(T,u))\\ &= \mathsf{lag}(T,t') + \mathsf{A}(\mathsf{ideal},T,t',t+1) - \mathsf{A}(\mathcal{S},T,t',t+1). & (15)\end{aligned}$$

Another useful definition, the total lag for a task system τ in a schedule $\mathcal{S}$ at time t, $\mathsf{LAG}(\tau,t,\mathcal{S})$, or more concisely, $\mathsf{LAG}(\tau,t)$, is given by

$$\mathsf{LAG}(\tau,t) = \textstyle\sum_{T\in\tau}\mathsf{lag}(T,t). \qquad (16)$$

Using (14)–(16), $\mathsf{LAG}(\tau,t+1)$ can be expressed as follows. In (18) below, $0 \le t' \le t$ holds.

$$\begin{aligned}\mathsf{LAG}(\tau,t+1) &= \mathsf{LAG}(\tau,t) + \textstyle\sum_{T\in\tau}(\mathsf{A}(\mathsf{ideal},T,t) - \mathcal{S}(T,t)) & (17)\\ \mathsf{LAG}(\tau,t+1) &= \mathsf{LAG}(\tau,t') + \sum_{u=t'}^{t}\sum_{T\in\tau}(\mathsf{A}(\mathsf{ideal},T,u) - \mathsf{A}(\mathcal{S},T,u))\\ &= \mathsf{LAG}(\tau,t') + \mathsf{A}(\mathsf{ideal},\tau,t',t+1) - \mathsf{A}(\mathcal{S},\tau,t',t+1) & (18)\end{aligned}$$

(17) and (18) above can be rewritten as follows using (7).

$$\begin{aligned}\mathsf{LAG}(\tau,t+1) &\le \mathsf{LAG}(\tau,t) + \textstyle\sum_{T\in\tau}(wt(T) - \mathcal{S}(T,t)) & (19)\\ \mathsf{LAG}(\tau,t+1) &\le \mathsf{LAG}(\tau,t') + (t+1-t')\cdot\textstyle\sum_{T\in\tau}wt(T) - \mathsf{A}(\mathcal{S},\tau,t',t+1) & (20)\\ &= \mathsf{LAG}(\tau,t') + (t+1-t')\cdot\textstyle\sum_{T\in\tau}wt(T) - \sum_{u=t'}^{t}\sum_{T\in\tau}\mathcal{S}(T,u) & (21)\end{aligned}$$

Soft real-time model. In *soft* real-time systems, tasks may miss their deadlines. As discussed in the introduction, this paper is concerned with deriving a lateness or *tardiness* [8] bound for a GIS task system scheduled under EPDF (described below). Formally, the tardiness of a subtask T_i in schedule $\mathcal{S}$ is defined as $tardiness(T_i,\mathcal{S}) = \max(0, t - \mathsf{d}(T_i))$, where t is the time at which T_i completes executing in $\mathcal{S}$. The tardiness of a task system τ under scheduling algorithm $\mathcal{A}$ is defined as the maximum tardiness of any subtask of any task in τ in any schedule for any concrete instantiation of τ under $\mathcal{A}$. If $\kappa(M)$ is the maximum tardiness of any task system with $U_{sum} \le M$ under $\mathcal{A}$ on M processors, then $\mathcal{A}$ is said to *ensure a tardiness bound of* $\kappa(M)$ on M processors. Though tasks in a soft real-time system are allowed to have nonzero tardiness, it is assumed that *missed deadlines do not delay future job releases*. Hence, guaranteeing a reasonable bound on tardiness that is independent of time is sufficient to ensure that in the long run each task is allocated a processor share that is in accordance with its weight. Because each task is sequential and subtasks of a task have an implicit precedence relationship, a later subtask cannot commence execution until all prior subtasks of the same task have completed execution. Thus, a missed deadline effectively reduces the interval over which the next subtask should be scheduled in order to meet its deadline.

Algorithm EPDF. Like most Pfair scheduling algorithms, the *earliest-pseudo-deadline-first* (EPDF) Pfair algorithm functions by choosing subtasks for execution at the beginning of every slot. Under EPDF, higher priority is accorded to subtasks with earlier deadlines; ties among subtasks with equal deadlines are resolved arbitrarily. In prior work, Srinivasan and Anderson have shown that EPDF is optimal on at most two processors [3]. They have also shown that on more than two processors, EPDF can correctly schedule task systems in which the maximum task weight is at most $1/(M-1)$ [12], and that EPDF can ensure a tardiness bound of $q \geq 1$ if the weight of each task is restricted to $\frac{q}{q+1}$ [11]. (Since EPDF is optimal on two processors, in deriving tardiness bounds under this algorithm, we assume that $M \geq 3$ holds.)

The above is a fairly comprehensive summary of basic Pfair scheduling. The rest of this section presents some additional definitions and results that we will use in this paper.

Active tasks. If subtasks are absent or are released late, then it is possible for a GIS (or IS) task to have no eligible subtasks and an allocation of zero during certain time slots. Tasks with and without subtasks in the interval $[t, t+\ell)$ are distinguished using the following definition of an *active* task.

Definition 1: A GIS task U is *active* in slot t if it has one or more subtasks U_j such that $\mathsf{e}(U_j) \leq t < \mathsf{d}(U_j)$. (A task that is active in t is not necessarily scheduled in that slot.)

Holes. If fewer than M tasks are scheduled at time t in $\mathcal{S}$, then one or more processors are idle in slot t. For each slot, each processor that is idle in that slot is referred to as a *hole*. Hence, if k processors are idle in slot t, then there are said to be k holes in t. The following lemma is a generalization of one proved in [10], and relates an increase in the total lag of τ, LAG, to the presence of holes.

Lemma 4. (*Srinivasan and Anderson [10]*) *If* $\mathsf{LAG}(\tau, t+\ell, \mathcal{S}) > \mathsf{LAG}(\tau, t, \mathcal{S})$*, where* $\ell \geq 1$*, then there is at least one hole in the interval* $[t, t+\ell)$.

Intuitively, if there is no idle processor in slots $t, \ldots, t+\ell-1$, then the total allocation in $\mathcal{S}$ in each of those slots to tasks in τ is equal to M. Since τ is assumed to be feasible, this is at least the total allocation that τ receives in any slot in the ideal schedule. Therefore, LAG cannot increase.

Task classification (from [10]). Tasks in τ may be classified as follows with respect to a schedule $\mathcal{S}$ and time interval $[t, t+\ell)$. (For brevity, we let the task system τ and schedule $\mathcal{S}$ be implicit in these definitions.)

$\boldsymbol{A(t, t+\ell)}$**:** Set of all tasks that are scheduled in one or more slots in $[t, t+\ell)$.

$\boldsymbol{B(t, t+\ell)}$**:** Set of all tasks that are not scheduled in any slot in $[t, t+\ell)$, but are active in one or more slots in the interval.

$\boldsymbol{I(t, t+\ell)}$**:** Set of all tasks that are neither active nor scheduled in any slot in $[t, t+\ell)$.

As a shorthand, the notation $A(t)$, $B(t)$, and $I(t)$ is used when $\ell = 1$. $A(t, t+\ell)$, $B(t, t+\ell)$, and $I(t, t+\ell)$ form a partition of τ, *i.e.*, the following holds.

$$A(t, t+\ell) \cup B(t, t+\ell) \cup I(t, t+\ell) = \tau \tag{22}$$

$$A(t, t+\ell) \cap B(t, t+\ell) = B(t, t+\ell) \cap I(t, t+\ell) = I(t, t+\ell) \cap A(t, t+\ell) = \emptyset \tag{23}$$

This classification of tasks is illustrated in Figure 4(a) for $\ell = 1$. Using (16), (22), and (23) above, we have the following.

$$\mathsf{LAG}(\tau, t+1) = \sum_{T \in A(t)} \mathsf{lag}(T, t+1) + \sum_{T \in B(t)} \mathsf{lag}(T, t+1) + \sum_{T \in I(t)} \mathsf{lag}(T, t+1) \tag{24}$$

The next definition identifies the last-released subtask at t of any task U.

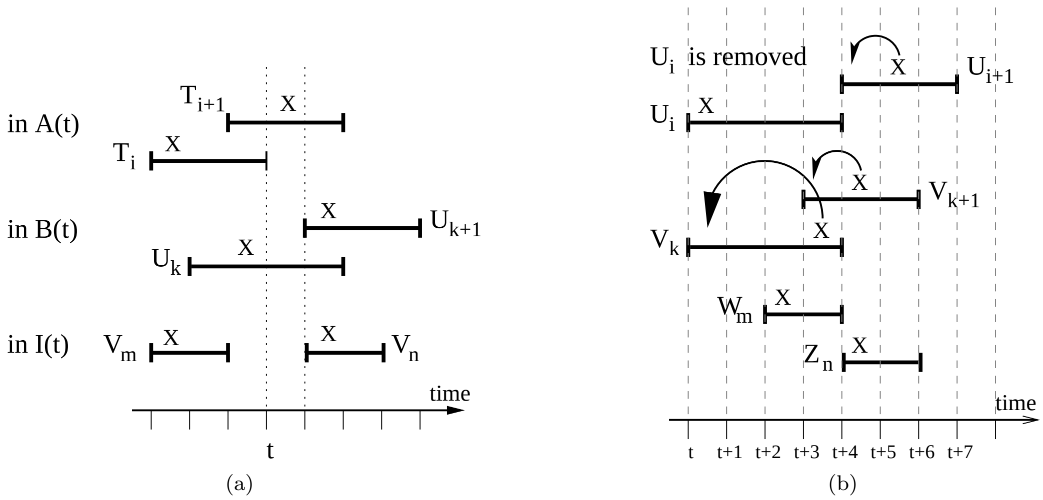

Figure 4: **(a)** Task classification at time t. IS-windows of two consecutive subtasks of three GIS tasks T, U, and V are depicted. The slot in which each subtask is scheduled is indicated by an "X." Because subtask T_{i+1} is scheduled at t, $T \in A(t)$. No subtask of U is scheduled at t. However, because the window of U_k overlaps slot t, U is active at t, and hence, $U \in B(t)$. Task V is neither scheduled at t, nor is it active at t. Therefore, $V \in I(t)$. **(b)** Illustration of displacements. If U_i, scheduled at time t, is removed from the task system, then some subtask that is eligible at t, but scheduled later, can be scheduled at t. In this example, it is subtask V_k (scheduled at $t+3$). This displacement of V_k results in two more displacements, those of V_{k+1} and U_{i+1}, as shown. Thus, there are three displacements in all: $\Delta_1 = (U_i, t, V_k, t+3), \Delta_2 = (V_k, t+3, V_{k+1}, t+4)$, and $\Delta_3 = (V_{k+1}, t+4, U_{i+1}, t+5)$.

Definition 2:. Subtask U_j is the *critical subtask of* U *at* t iff $\mathsf{e}(U_j) \leq t < \mathsf{d}(U_j)$ holds, and no other subtask U_k of U, where $k > j$, satisfies $\mathsf{e}(U_k) \leq t < \mathsf{d}(U_k)$. For example, in Figure 4(a), the critical subtask of T at both $t-1$ and t is T_{i+1}, and that of U at $t+1$ is U_{k+1}.

Displacements. To facilitate reasoning about Pfair algorithms, Srinivasan and Anderson formally defined displacements in [10]. Let τ be a GIS task system and let $\mathcal{S}$ be an EPDF schedule for τ. Then, removing a subtask, say T_i, from τ results in another GIS task system τ'. Suppose T_i is scheduled at t in $\mathcal{S}$. Then, T_i's removal can cause another subtask, say U_j, scheduled after t to shift left to t, which in turn can lead to other shifts, resulting in an EPDF schedule $\mathcal{S}'$ for τ'. Each shift that results due to a subtask removal is called a *displacement* and is denoted by a four-tuple $\langle X^{(1)}, t_1, X^{(2)}, t_2 \rangle$, where $X^{(1)}$ and $X^{(2)}$ represent subtasks. This is equivalent to saying that subtask $X^{(2)}$ originally scheduled at t_2 in $\mathcal{S}$ displaces subtask $X^{(1)}$ scheduled at t_1 in $\mathcal{S}$. A displacement $\langle X^{(1)}, t_1, X^{(2)}, t_2 \rangle$ is *valid* iff $\mathsf{e}(X^{(2)}) \leq t_1$. Because there can be a cascade of shifts, there may be a chain of displacements. Such a chain is represented by a sequence of four-tuples. An example is given in Figure 4(b).

The next lemma regarding displacements is proved in [9]. It states that in an EPDF schedule, a subtask removal can cause other subtasks to shift only to their left.

Lemma 5. (*from* [9]) *Let* $X^{(1)}$ *be a subtask that is removed from* τ*, and let the resulting chain of displacements in an* EPDF *schedule for* τ *be* $C = \Delta_1, \Delta_2, \ldots, \Delta_k$*, where* $\Delta_i = \langle X^{(i)}, t_i, X^{(i+1)}, t_{i+1} \rangle$*. Then* $t_{i+1} > t_i$ *for all* $i \in [1, k]$*.*

3. Tardiness Bounds for EPDF

In this section, we present results concerning tardiness bounds that can be guaranteed under EPDF.

Table 1: Counterexamples to show that tardiness under EPDF can exceed three.

	Task Set		Util. (M)	Tardiness (in quanta)
	# of tasks	weight		
τ_1	4	1/2	10	2 at 50
	3	3/4		
	6	23/24		
τ_2	4	1/2	19	3 at 963
	3	3/4		
	5	23/24		
	10	239/240		
τ_3	4	1/2	80	4 at 43,204
	3	3/4		
	3	23/24		
	1	31/32		
	4	119/120		
	4	239/240		
	6	479/480		
	8	959/960		
	15	1199/1200		
	15	2399/2400		
	20	4799/4800		

It is easy to show that subtask deadlines can be missed under EPDF. In [11], it was conjectured that EPDF ensures a tardiness of at most one for every feasible task system. We now show that this conjecture is false.

Theorem 1. *Tardiness under* EPDF *can exceed three quanta for feasible GIS task systems. In particular, if* EPDF *is used to schedule task system* τ_i $(1 \leq i \leq 3)$ *in Table 1, then a tardiness of* $i+1$ *quanta is possible.*

Proof: Figure 5 shows a schedule for τ_1, in which a subtask has a tardiness of two at time 50. The schedules for τ_2 and τ_3 are too lengthy to be depicted; we verified them using two independently-coded EPDF simulators. ■

The sufficient condition for a tardiness of $q > 0$ quanta as given by Srinivasan and Anderson requires that the sum of the weights of the $M-1$ heaviest tasks be less than $\frac{qM+1}{q+1}$. This can be ensured if the weight of each task is restricted to be at most $\frac{q}{q+1}$. We next show that a weight restriction of $\frac{q+2}{q+3}$ per task is sufficient to guarantee a tardiness of q quanta. This restriction is stated below.

(W) The weight of each task is at most $\frac{q+2}{q+3}$.

In what follows, we prove the following theorem.

Theorem 2. *Tardiness under* EPDF *is at most* q *quanta, where* $q \geq 1$*, for every GIS task system that is feasible on* $M \geq 3$ *processors and satisfies* (W).

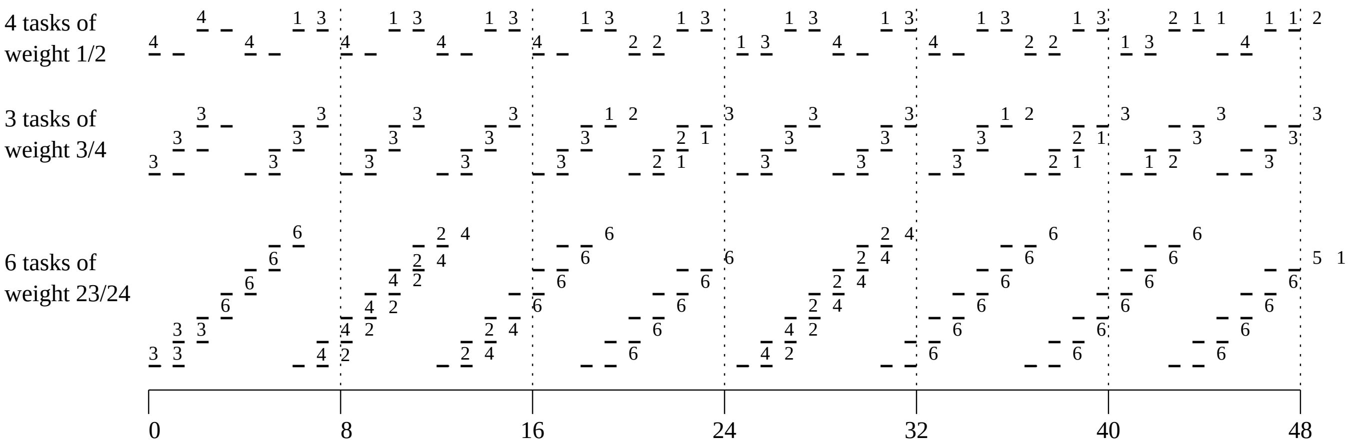

Figure 5: Counterexample to prove that tardiness under EPDF can exceed one quantum. 13 periodic tasks with total utilization ten are scheduled on ten processors. In the schedule, tasks of the same weight are shown together as a group. Each column corresponds to a time slot. The PF-window of each subtask is shown as a sequence of dashes that are aligned. An integer value n in slot t means that n tasks in the corresponding group have a subtask scheduled at t. Subtasks that miss deadlines are shown scheduled after their windows. In this schedule, 11 subtasks miss their deadlines at time 48. Hence, tardiness is 2 quanta for at least one subtask.

We use a setup similar to that used by Srinivasan and Anderson in [10] and [11] to prove the above theorem. Though the setup is similar and some fundamental properties are applicable, there are significant differences in the core of the proof.

Our proof is by contradiction, hence, assume Theorem 2 does not hold. This assumption implies that there exists a $q \geq 1$, a time t_d, and a concrete task system σ defined as follows.

Definition 3:. t_d is the earliest deadline of a subtask with a tardiness of $q+1$ under EPDF in any feasible GIS task system satisfying (W), *i.e.*, there exists some such task system with a subtask with deadline at t_d and tardiness $q+1$, and there does not exist any such task system with a subtask with deadline prior to t_d and a tardiness of $q+1$.

Definition 4:. σ is a feasible concrete GIS task system satisfying (W) with the following properties.

(**S1**) A subtask in σ with deadline at t_d has a tardiness of $q+1$ under EPDF.

(**S2**) No feasible concrete task system satisfying (W) and (S1) releases fewer subtasks in $[0, t_d)$ than σ.

In what follows, let $\mathcal{S}'$ denote an EPDF schedule for σ in which a subtask of σ with deadline at t_d has a tardiness of $q+1$.

By (S1) and (S2), exactly one subtask in σ has a tardiness of $q+1$: if several such subtasks exist, then all but one can be removed and the remaining subtask will still have a tardiness of $q+1$, contradicting (S2). Similarly, a subtask with deadline later than t_d cannot impact how subtasks with deadlines at or before t_d are scheduled. Therefore, no subtask in σ has a deadline after t_d. Based on these facts, Lemma 6 below can be shown to hold. In proving Lemma 6, we use the following claim, proved in an appendix.

Claim 1. *There is no hole in any slot in* $[t_d - 1, t_d + q)$ *in* $\mathcal{S}'$.

We now show that LAG of σ at t_d is exactly $qM+1$.

Lemma 6. $\mathsf{LAG}(\sigma, t_d, \mathcal{S}') = qM + 1$.

Proof: By Claim 1, there is no hole in any slot in $[t_d, t_d + q)$ in $\mathcal{S}'$. Further, the subtask with a tardiness of $q+1$ and deadline at t_d, as specified in (S1), is not scheduled until time $t_d + q$. (Also, recall that there is exactly one such subtask.) Thus, because every subtask in σ has a deadline of at most t_d, there exist exactly $qM+1$ subtasks with deadlines at most t_d that are pending at t_d in $\mathcal{S}'$. In the ideal schedule, all of these subtasks complete executing by time t_d. Therefore, the LAG of σ at t_d, which is the difference between the ideal allocation and the allocation in $\mathcal{S}'$ in $[0, t_d)$, is $qM+1$. ■

By Claim 1, there is no hole in slot t_d-1. Therefore, by the contrapositive of Lemma 4, $\mathsf{LAG}(\sigma, t_d-1, \mathcal{S}') \geq \mathsf{LAG}(\sigma, t_d, \mathcal{S}')$, which, by Lemma 6, is $qM+1$. Thus, because $\mathsf{LAG}(\sigma, 0, \mathcal{S}') = 0$, there exists a time t, where $0 \leq t < t_d - 1$ such that $\mathsf{LAG}(\sigma, t, \mathcal{S}') < qM+1$ and $\mathsf{LAG}(\sigma, t+1, \mathcal{S}') \geq qM+1$. This further implies the existence of a time $0 \leq t_h < t_d - 1$, a concrete task system τ, and an EPDF schedule $\mathcal{S}$ for τ defined as follows.

Definition 5:. t_h, where $0 \leq t_h < t_d - 1$, is the earliest time such that the LAG in any EPDF schedule for any feasible concrete GIS task system satisfying (W) is at least $qM+1$ at t_h+1.

Definition 6:. τ is a feasible concrete GIS task system satisfying (W) with the following properties.

(**T1**) $\mathsf{LAG}(\tau, t_h+1, \mathcal{S}) \geq qM+1$.

(**T2**) No feasible concrete task system satisfying (W) and (T1) releases fewer subtasks than τ.

(**T3**) No feasible concrete task system satisfying (W), (T1), and (T2) has a larger rank than τ where the *rank* of a task system is the sum of the eligibility times of all its subtasks, *i.e.*, $rank(\tau, t) = \sum_{\{T_i \in \tau\}} \mathsf{e}(T_i)$.

(T2) can be thought of as identifying a minimal task system in the sense of having LAG exceed $qM+1$ at the earliest possible time with the fewest number of subtasks, subject to satisfying (W). As already explained, if Theorem 2 does not hold for all task systems satisfying (W), then there exists some task system whose LAG is at least $qM+1$. Therefore, some task system satisfying (W), (T1), and (T2) necessarily exists. (T3) further restricts the nature of such a task system by requiring subtask eligibility times to be spaced as much apart as possible.

We next prove some properties about the subtasks of τ scheduled in $\mathcal{S}$.

Lemma 7. *Let T_i be a subtask in τ. Then, the following properties hold for T_i in $\mathcal{S}$.*

(**a**) *If T_i is scheduled at t, then $\mathsf{e}(T_i) \geq \min(\mathsf{r}(T_i), t)$.*

(**b**) *If T_i is scheduled before t_d, then the tardiness of T_i is at most q.*

Proof of part (a). Suppose $\mathsf{e}(T_i)$ is not equal to $\min(\mathsf{r}(T_i), t)$. Then, by (3) and because T_i is scheduled at t, it is before $\min(\mathsf{r}(T_i), t)$. Hence, simply changing $\mathsf{e}(T_i)$ so that it equals $\min(\mathsf{r}(T_i), t)$ will not affect how T_i or any other subtask is scheduled. Therefore, the actual allocations in $\mathcal{S}$ to every task, and hence, the lag of every task, will remain unchanged. Therefore, the LAG of τ at t_h+1 will still be at least $qM+1$. However, changing the eligibility time of T_i increases the rank of the task system, and hence, (T3) is contradicted. ■

Proof of part (b). Follows from Definition 3. ■

In what follows, we show that if (W) is satisfied, then there does not exist a time t_h as defined in Definition 5, that is, we contradict its existence, and in turn prove Theorem 2. For this, we deduce the LAG of τ at t_h+1 by determining the lags of the tasks in τ. But first, a brief digression on subtask categorization that will help improve the accuracy with which task lags are bound.

3.1. Categorization of Subtasks

As can be seen from (13) and (6), the lag of a task T at t depends on the allocations that subtasks of T receive in each time slot until t in the ideal schedule. Hence, a tight estimate of such allocations is essential to bounding the lag of T reasonably accurately. If a subtask's index is not known, then (6), which can otherwise be used to compute the allocation received by any subtask in any slot *exactly*, is not of much help. Hence, in this subsection, we define terms that will help in categorizing subtasks, and then derive upper bounds for the allocations that these categories of subtasks receive in certain slots in the ideal schedule.

***k*-dependent subtasks.** The subtasks of a heavy task with weight in the range $[1/2, 1)$ can be divided into "groups" based on their group deadlines in a straightforward manner: place all subtasks with identical group deadlines in the same *group* and identify the group using the smallest index of any subtask in that group. For example, in Figure 2, $G_1 = \{T_1, T_2\}$, $G_3 = \{T_3, T_4, T_5\}$, and $G_6 = \{T_6, T_7, T_8\}$. If there are no IS separations or GIS omissions among the subtasks of a group, then a deadline miss by q quanta for a subtask T_i will necessarily result in a deadline miss by at least q quanta for the subsequent subtasks in T_i's group. Hence, a subtask T_j is dependent on all prior subtasks in its group for not missing its deadline. If T is heavy, we say that T_j is *k-dependent*, where $k \geq 0$, if T_j is the $(k+1)^{\text{st}}$ subtask in its group, computed assuming all subtasks are present (that is, as in the determinination of group deadlines, even if T is GIS and some subtasks are omitted, k-dependency is determined assuming there are no omissions).

Recall that by Lemma 1, all subtasks of a heavy task with weight less than one are of length two or three. Further, note that in each group, each subtask except possibly the first is of length two. This implies that for a periodic task the deadlines of any two successive subtasks that belong to the same group differ by exactly one. Also, in each group, each subtask except possibly the final subtask has a b-bit of one. Finally, if the final subtask of a group has a b-bit of one, then the first subtask of the group that follows is of length three. These properties are summarized in the following lemma.

Lemma 8. *The following properties hold.*

(a) *Let T be a heavy task with $wt(T) < 1$ and let T_i be a 0-dependent subtask of T. Then, one of the following holds:* **(i)** *$i = 1$;* **(ii)** *$b(T_{i-1}) = 0$;* **(iii)** *$|\omega(T_i)| = 3$.*

(b) *If T_i is a k-dependent subtask of a **periodic** task T, where $i \geq 2$ and $k \geq 1$, then $d(T_i) = d(T_{i-1}) + 1$ and $r(T_i) = d(T_{i-1}) - 1$.*

(c) *Let T_i, where $i > 1$, be a k-dependent subtask of T with $wt(T) < 1$. If $k \geq 1$, then $|\omega(T_i)| = 2$ and $b(T_{i-1}) = 1$.*

If a task T is light, then we simply define all of its subtasks to be 0-dependent. In this case, each subtask is in its own group.

Miss initiators. A subtask scheduled at t and missing its deadline by c quanta, where $c \geq 1$, is referred to as a *miss initiator by c* (or a c-MI, for short) for its group, if no subtask of the same task is scheduled at $t-1$. (A miss initiator by q, *i.e.*, a q-MI, will simply be referred to as an MI.) Thus, a subtask is a c-MI if it misses its deadline by c quanta and is either the first subtask in its group to do so or separated from its predecessor by an IS or GIS separation, and its predecessor is not scheduled in the immediately preceding slot. Such a subtask is termed a miss initiator by c because in the absence of future separations, it causes all subsequent subtasks in its group to miss their deadlines by c quanta as well. Several examples of MIs for $q = 1$ are shown in Figure 6.

Successors of miss initiators. The immediate successor T_{i+1} of a c-MI T_i is called a *successor of a c-MI* (or c-SMI, for short) if $tardiness(T_{i+1}) = tardiness(T_i) = c$, $\mathcal{S}(T_{i+1}, t) = 1 \Rightarrow \mathcal{S}(T_i, t-1) = 1$, and T_i is a c-MI. (A successor of a miss initiator by q, *i.e.*, a q-SMI, will simply be referred to as an SMI.) Figure 6

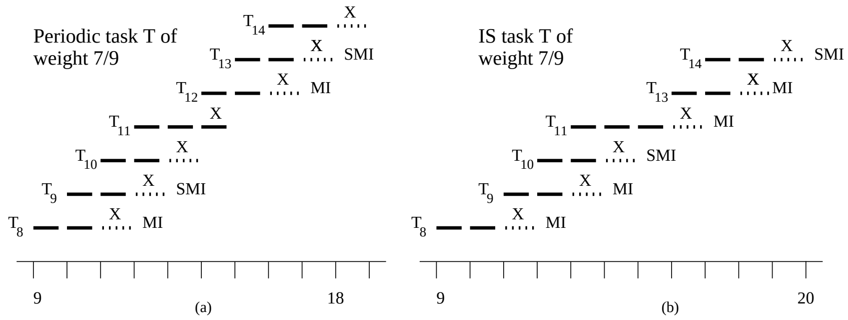

Figure 6: Possible schedules for the second job of **(a)** a periodic and **(b)** a GIS task of weight 7/9 under EPDF. Subtasks are scheduled in the slots marked by an X. Solid (dotted) lines indicate slots that lie within (outside) the window of a subtask. A subtask scheduled in a dotted slot misses its deadline. In (a), T_8 and T_{12} are MIs, T_9 and T_{13} are SMIs, and the remaining subtasks fall within neither category. T_{10} and T_{14} have a tardiness of one, and T_{11} has a tardiness of zero. In (b), T_8, T_9, T_{11}, and T_{13} are MIs, and T_{10} and T_{14} are SMIs. Note that T_8 and T_9 (T_{11} and T_{13}) belong to the same group G_8 (G_{11}). Thus, if there are IS separations, there may be more than one MI in a group.

shows several examples for $q = 1$. Note that for T_{i+1} to be a c-SMI, its predecessor in $\mathcal{S}$ must be T_i, rather than some lower-indexed subtask of T. Also note that a c-SMI is at least 1-dependent.

The lemma below follows immediately from Lemma 8(a), which by (1) implies that the deadline of the first subtask of a group is greater than that of the final subtask of the preceding group by at least two.

Lemma 9. *Let T_i be a subtask that is scheduled at t and let T_i's tardiness be $c > 0$ quanta. If T_j, where $j < i$, is scheduled at $t - 1$ and T_j does not belong to the same dependency group as T_i, then the tardiness of T_j is at least $c + 1$.*

The next lemma bounds the allocation received by a k-dependent subtask in the first slot of its window in the ideal schedule, and is proved in an appendix.

Lemma 10. *The allocation received by a k-dependent subtask in its first slot in the ideal schedule are as follows.*

(a) *The allocation $\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{r}(T_i))$ received in the ideal schedule by a k-dependent subtask T_i of a* ***periodic*** *task T with $wt(T) < 1$ in the first slot of its window is at most $k \cdot \frac{T.e}{T.p} - (k-1) - \frac{1}{T.p}$, for all $k \geq 0$.*

(b) *The allocation $\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{r}(T_i))$ received in the ideal schedule by a k-dependent subtask T_i of a GIS task T in the first slot of its window is at most $k \cdot \frac{T.e}{T.p} - (k-1) - \frac{1}{T.p}$, for all $k \geq 0$.*

(c) *Let T_i, where $i \geq k+1$ and $k \geq 1$, be a subtask of T with $wt(T) < 1$ such that $|\omega(T_i)| \geq 3$ and $b(T_{i-1}) = 1$. Let the number of subtasks in T_{i-1}'s dependency group be at least k. Then, $\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{r}(T_i)) \leq k \cdot \frac{T.e}{T.p} - (k-1) - \frac{1}{T.p}$.*

The next lemma bounds the lag of a task at time t, based on the k-dependency of its last-scheduled subtask. This is also proved in an appendix.

Lemma 11. *Let T_i be a k-dependent subtask of a task T for $k \geq 0$, and let the tardiness of T_i be s for some $s \geq 1$ (that is, T_i is scheduled at time $\mathsf{d}(T_i) + s - 1$). Then $\mathsf{lag}(T, \mathsf{d}(T_i) + s) < (k + s + 1) \cdot wt(T) - k$.*

3.2. Subclassification of Tasks in $A(t)$

Recall from Section 2 that a task in $A(t)$ is scheduled in slot t. We further classify tasks in $A(t)$, based on the tardiness of their subtasks scheduled at t, as follows.

$\boldsymbol{A_0(t)}$**:** Includes T in $A(t)$ iff its subtask scheduled at t has zero tardiness.

$\boldsymbol{A_q(t)}$**:** Includes T in $A(t)$ iff its subtask scheduled at t has a tardiness of q.

$\boldsymbol{A_{q-1}(t), q > 1}$**:** Includes T in $A(t)$ iff its subtask scheduled at t has a tardiness greater than 0 but less than q.

$A_q(t)$ is further partitioned into $A_q^0(t)$, $A_q^1(t)$, and $A_q^2(t)$.

$\boldsymbol{A_q^0(t)}$**:** Includes T in $A_q(t)$ iff its subtask scheduled at t is an MI.

$\boldsymbol{A_q^1(t)}$**:** Includes T in $A_q(t)$ iff its subtask scheduled at t is an SMI.

$\boldsymbol{A_q^2(t)}$**:** Includes T in $A_q(t)$ iff its subtask scheduled at t is neither an MI nor an SMI.

$\boldsymbol{A_{q-1}^0(t), q > 1}$**:** Includes T in $A_{q-1}(t)$ iff its subtask scheduled at t is a c-MI, where $0 < c < q$.

From the above, we have the following.

$$A_0(t) \cup A_q(t) \cup A_{q-1}(t) = A(t) \text{ and } A_q^0(t) \cup A_q^1(t) \cup A_q^2(t) = A_q(t) \tag{25}$$
$$A_0(t) \cap A_q(t) = A_q(t) \cap A_{q-1}(t) = A_{q-1}(t) \cap A_0(t) = \emptyset \tag{26}$$
$$A_q^0(t) \cap A_q^1(t) = A_q^1(t) \cap A_q^2(t) = A_q^2(t) \cap A_q^0(t) = \emptyset \tag{27}$$

.

3.3. Task Lags by Task Classes and Subclasses

By the definition there of t_h in Definition 5, $\mathsf{LAG}(\tau, t_h + 1) > \mathsf{LAG}(\tau, t_h)$. Hence, by Lemma 4, the following holds.

(H) There is at least one hole in slot t_h.

The next lemma gives bounds on the lags of tasks in $A(t)$, $B(t)$, and $I(t)$ at $t + 1$, where $t \leq t_h$ is a slot with a hole, and hence, hold the lemma holds for $t = t_h$, as well.

Lemma 12. *Let $t \leq t_h$ be a slot with a hole. Then, the following bounds hold for* lag *at $t + 1$ of a task T depending on whether it is scheduled at t and the type of its subtask scheduled at that time.*

(a) *(from [11]) For* $T \in I(t), \mathsf{lag}(I, t + 1) = 0$.

(b) *(from [11]) For* $T \in B(t), \mathsf{lag}(B, t + 1) \leq 0$.

(c) *(from [6]) For* $T \in A_0(t), \mathsf{lag}(T, t + 1) < wt(T)$.

(d) *For* $T \in A_q^0(t)$, $\mathsf{lag}(T, t + 1) < (q + 1) \cdot wt(T)$.

(e) *For* $T \in A_q^1(t)$, $\mathsf{lag}(T, t + 1) < (q + 2) \cdot wt(T) - 1$.

(f) *For $T \in A_q^2(t)$, $\mathsf{lag}(T, t+1) < (q+3) \cdot wt(T) - 2$.*

(g) *For $T \in A_{q-1}(t)$, $\mathsf{lag}(T, t+1) < q \cdot wt(T)$.*

Proof: parts (a) and (b) are proved in [11]. To see why they hold, note that no task in $B(t)$ or $I(t)$ is scheduled at t. Because there is a hole in t, the critical subtask of a task in $B(t)$ is scheduled before t; similarly, the latest subtask of a task in $I(t)$ with release time at or before t should have completed execution by t. Hence, such tasks cannot be behind with respect to the ideal schedule. part (c) is proved in [6]. The remaining parts are proved below.

Proof of part (d). If $T \in A_q^0(t)$, then the subtask T_i of T scheduled at t is an MI, and $\mathsf{d}(T_i) = t - q + 1$. Further T_i is k-dependent, where $k \geq 0$. Hence, by Lemma 11, $\mathsf{lag}(T, t+1)$ is less than $(k+q+1) \cdot wt(T) - k$, which (because $wt(T) \leq 1$) is at most $(q+1) \cdot wt(T)$, for all $k \geq 0$. ■

Proof of part (e). If $T \in A_q^1(t)$, then the subtask T_i of T scheduled at t is an SMI, and is k-dependent for $k \geq 1$. Also, $\mathsf{d}(T_i) = t - q + 1$. Thus, by part (11), $\mathsf{lag}(T, t+1) < (k+q+1) \cdot wt(T) - k \leq (q+2) \cdot wt(T) - 1$ for all $k \geq 1$ (because $wt(T) \leq 1$). ■

Proof of part (f). Similar to that of part (e). ■

Proof of part (g). Let T_i be T's subtask scheduled at t and let s denote the tardiness of T_i. Then, $t + 1 = \mathsf{d}(T_i) + s$. Let T_i be k-dependent, where $k \geq 0$. By the definition of A_{q-1}, $0 < s < q$ holds, and by Lemma 11, $\mathsf{lag}(T, \mathsf{d}(T_i) + s) = \mathsf{lag}(T, t+1) < (k+s+1) \cdot wt(T) - k \leq (k+q) \cdot wt(T) - k \leq q \cdot wt(T)$, for all $k \geq 0$. ■

3.4. Some Auxiliary Lemmas

In proving Theorem 2, we also make use of the following three lemmas, established in prior work by Srinivasan and Anderson.

Lemma 13. (*Srinivasan and Anderson* [10]) *If $\mathsf{LAG}(\tau, t+1) > \mathsf{LAG}(\tau, t)$, then $B(t) \neq \emptyset$.*

The following is an intuitive explanation for why Lemma 13 holds. Recall from Section 2 that $B(t)$ is the set of all tasks that are active but not scheduled at t. Because $\mathsf{e}(T_i) \leq \mathsf{r}(T_i)$ holds, by Definition 1 and (6), only tasks that are active at t may receive positive allocations in slot t in the ideal schedule. Therefore, if every task that is active at t is scheduled at t, then the total allocation in $\mathcal{S}$ cannot be less than the total allocation in the ideal schedule, and hence, by (17), LAG cannot increase across slot t.

Lemma 14. (*Srinivasan and Anderson* [10]) *Let $t < t_d$ be a slot with holes and let $T \in B(t)$. Then, the critical subtask at t of T is scheduled before t.*

To see that the above lemma holds, let T_i be the critical subtask of T at t. By its definition, the IS-window of T_i overlaps slot t, but T is not scheduled at t. Also, there is at least a hole in t. Because EPDF does not idle a processor while there is a task with an outstanding execution request, T_i is thus scheduled before t.

Lemma 15. (*Srinivasan and Anderson* [10]) *Let U_j be a subtask that is scheduled in slot t', where $t' \leq t \leq t_h$ and let there be a hole in t. Then, $\mathsf{d}(U_j) \leq t+1$.*

This lemma is true because it can be shown that if $\mathsf{d}(U_j) > t+1$ holds, then U_j has no impact on how subtasks are scheduled after t. In particular, it can be shown that even if U_j is removed, no subtask scheduled after t can be scheduled at or before t. Therefore, it can be shown that if the lemma does not hold, then the GIS task system obtained from τ by removing U_j also has a LAG at least $qM+1$ at t_h+1, which is a contradiction to (T2).

Arguments similar to those used in proving the above lemma can be used to show the following. This lemma is proved in [6]

Lemma 16. *(from [6]) Let $t < t_d$ be a slot with holes. Let U_j be a subtask that is scheduled at t and let the tardiness of U_j be zero. Then, $\mathsf{d}(U_j) = t+1$ and $b(U_j) = 1$.*

In the rest of this subsection, we will establish three more lemmas for later use. But first, a couple of definitions.

By Definition 5, $\mathsf{LAG}(\tau, t_h+1) > \mathsf{LAG}(\tau, t_h)$. Therefore, by Lemma 13, $B(t_h) \neq \emptyset$. By (H), there is at least one hole in t_h. Hence, by Lemma 14, the critical subtask at t_h of every task in $B(t_h)$ is scheduled before t_h. The next definition identifies the latest time at which a critical subtask at t_h of any task in $B(t_h)$ is scheduled.

Definition 7:. t_b denotes the latest time before t_h at which the subtask that is critical at t_h of any task in $B(t_h)$ is scheduled.

U and U_j are henceforth to be taken as defined below.

Definition 8 (U and U_j):. U denotes a task in $B(t_h)$ with a subtask U_j that is critical at t_h scheduled at t_b.

The lemma below shows that the deadline of the critical subtask at t_h of every task in $B(t_h)$ is at t_h+1.

Lemma 17. *Let T be a task in $B(t_h)$ and let T_i be T's critical subtask at t_h. Then, $\mathsf{d}(T_i) = t_h+1$.*

Proof: Because T is in $B(t_h)$, T is active at t_h, but is not scheduled at t_h. Hence, T_i, which is critical at t_h, should have been scheduled earlier. In this case, by Lemma 15, $\mathsf{d}(T_i) \leq t_h+1$ holds. However, since T_i is T's critical subtask at t_h, by Definition 2, $\mathsf{d}(T_i) \geq t_h+1$ holds. Therefore, $\mathsf{d}(T_i) = t_h+1$ follows. ■

The following lemma shows that at least one subtask scheduled in t_h has a tardiness of zero, *i.e.*, $|A_0(t_h)| \geq 1$. It is proved in an appendix.

Lemma 18. *There exists a subtask W_ℓ scheduled at t_h with $\mathsf{e}(W_\ell) \leq t_b$, $\mathsf{d}(W_\ell) = t_h+1$, and $\mathcal{S}(W,t) = 0$, for all $t \in [t_b, t_h)$. Also, there is no hole in any slot in $[t_b, t_h)$. (Note that, by this lemma, $A_0(t_h) \neq \emptyset$.)*

The next lemma establishes some properties with respect to a slot in which at least one MI is scheduled. It is also proved in an appendix.

Lemma 19. *Let $t_m \leq t_h$ be a slot in which an MI is scheduled. Then, the following hold.*

(a) *For all t, where $t_m - (q+2) < t < t_m$, there is no hole in slot t, and for each subtask V_k that is scheduled in t, $\mathsf{d}(V_k) \leq t_m - q + 1$.*

(b) *Let W be a task in $B(t_m)$ and let the critical subtask W_ℓ of W at t_m be scheduled before t_m. Then, W_ℓ is scheduled at or before $t_m - (q+2)$.*

3.5. Core of the Proof

Having classified the tasks at t_h and determined their lags at t_h+1, we next show that if (W) holds, then $\mathsf{LAG}(\tau, t_h+1) < M+1$ in each of the following cases.

For conciseness, in what follows, we denote subsets $A(t_h)$, $B(t_h)$, and $I(t_h)$ as A, B, and I, respectively. Subsets $A_{q-1}(t_h)$ and $A_q(t_h)$ and their subsets are similarly denoted without the time parameter.

Case A: $A_q = \emptyset$.

Case B: $A_q^0 \neq \emptyset$ or ($A_q^1 \neq \emptyset$ and $A_{q-1}^0 \neq \emptyset$).

Case C: $A_q^0 = \emptyset$ and $A_q^1 \neq \emptyset$ and $A_{q-1}^0 = \emptyset$.

Case D: $A_q^0 = A_q^1 = \emptyset$.

The following notation is used to denote subset cardinality.

$$a_0 = |A_0|;\ a_q = |A_q|;\ a_q^0 = |A_q^0|;\ a_q^1 = |A_q^1|;\ a_q^2 = |A_q^2|;$$
$$a_{q-1}^0 = |A_{q-1}^0|;\ a_{q-1} = |A_{q-1}|.$$

h is defined as follows. (There is no correspondence between h as defined here and the subscript h in t_h. The subscript h in t_h is just an indication that t_h is a slot with holes.)

$$h \stackrel{\text{def}}{=} \text{number of holes in } t_h$$

Because there is at least one hole in t_h

$$h > 0. \tag{28}$$

In the remainder of this paper, let $W_{\max}$ denote the maximum weight of any task in τ. That is,

$$W_{\max} = \max_{T\in\tau}\{wt(T)\}. \tag{29}$$

In each of the above cases, $\mathsf{LAG}(\tau, t_h+1)$ can be expressed as follows.

$$\begin{aligned}
\mathsf{LAG}(\tau, t_h+1) &= \sum_{T\in\tau} \mathsf{lag}(T, t_h+1) \\
&\leq \sum_{T\in A_0} \mathsf{lag}(T, t_h+1) + \sum_{T\in A_{q-1}} \mathsf{lag}(T, t_h+1) + \sum_{T\in A_q^0} \mathsf{lag}(T, t_h+1) + \\
&\quad \sum_{T\in A_q^1} \mathsf{lag}(T, t_h+1) + \sum_{T\in A_q^2} \mathsf{lag}(T, t_h+1) \quad \text{((by (22), (23), (25), and Lemmas 12(a) and (b)))} \\
&< \sum_{T\in A_0} wt(T) + \sum_{T\in A_{q-1}} q\cdot wt(T) + \sum_{T\in A_q^0} (q+1)\cdot wt(T) + \\
&\quad \sum_{T\in A_q^1} ((q+2)\cdot wt(T) - 1) + \sum_{T\in A_q^2} ((q+3)\cdot wt(T) - 2) \quad \text{(by Lemmas 12(c)–(g))}
\end{aligned}$$

Using (29), $\mathsf{LAG}(\tau, t_h+1)$ can be bounded as

$\mathsf{LAG}(\tau, t_h+1)$

$$
\begin{aligned}
< \quad & a_0 \cdot W_{\max} + a_{q-1} \cdot q \cdot W_{\max} + a_q^0 (q+1) W_{\max} + a_q^1 ((q+2) W_{\max} - 1) + \\
& a_q^2 ((q+3) W_{\max} - 2) \qquad (30) \\
\leq \quad & \begin{cases} a_0 \cdot W_{\max} + a_q^0 \cdot (q+1) W_{\max} + a_q^1 \cdot ((q+2) W_{\max} - 1) + \\ (a_{q-1} + a_q^2) \cdot ((q+3) W_{\max} - 2), & W_{\max} \geq \frac{2}{3} \\ \\ a_0 \cdot (2/3) + a_q^0 \cdot (q+1)(2/3) + a_q^1 \cdot ((q+2)(2/3) - 1) + \\ (a_{q-1} + a_q^2) \cdot (q \cdot (2/3)), & W_{\max} < \frac{2}{3} \end{cases} \qquad (31) \\
& \text{(because } (q+3) W_{\max} - 2 \geq q \cdot W_{\max} \text{ for } W_{\max} \geq 2/3\text{).}
\end{aligned}
$$

Note that though $(q+3)W_{\max} - 2 < q \cdot W_{\max}$ holds, for $W_{\max} < 2/3$, $(q+3) \cdot (2/3) - 2 = (2/3) \cdot q > q \cdot W_{\max}$ holds for all $W_{\max} < 2/3$. Therefore, if the values of a_0, a_{q-1}, and a_q^i are not dependent on whether $W_{\max} \geq 2/3$ or $W_{\max} < 2/3$, determining a bound on $\mathsf{LAG}(\tau, t_h + 1)$ using the expression corresponding to $W_{\max} \geq 2/3$ in (31) (of course, assuming that $W_{\max} \geq 2/3$) serves as an upper bound for LAG when $W_{\max} < 2/3$. Hence, later in the paper, when a_0, a_{q-1}, and a_q^i are not dependent on $W_{\max}$, we bound $\mathsf{LAG}(\tau, t_h + 1)$ in this way.

The total number of processors, M, expressed in terms of the number of subtasks in each subset of A scheduled at t_h, and the number of holes in t_h, is as follows.

$$M = a_0 + a_{q-1} + a_q^0 + a_q^1 + a_q^2 + h \qquad (32)$$

3.6. Case A: $A_q = \emptyset$

Case A is dealt with as follows.

Lemma 20. *If* $A_q = \emptyset$*, then* $\mathsf{LAG}(\tau, t_h + 1) < qM + 1$.

Proof: If $A_q = \emptyset$, then

$$
\begin{aligned}
& \mathsf{LAG}(\tau, t_h + 1) \\
& \quad < \quad a_0 \cdot W_{\max} + a_{q-1} \cdot q \cdot W_{\max} && \text{(by (30) and } a_q^0 = a_q^1 = a_q^2 = 0\text{)} \\
& \quad \leq \quad a_0 \cdot q \cdot W_{\max} + a_{q-1} \cdot q \cdot W_{\max} \\
& \quad < \quad (M - h) \cdot q \cdot W_{\max} && \text{(by (32), } a_0 + a_{q-1} = M - h \text{ for this case)} \\
& \quad < \quad qM + 1. && \blacksquare
\end{aligned}
$$

Hence, if no subtask with a tardiness of q is scheduled in t_h, then (T1) is contradicted.

3.7. Case B: $A_q^0 \neq \emptyset$ *or* ($A_q^1 \neq \emptyset$ *and* $A_{q-1}^0 \neq \emptyset$)

By Lemma 12(d), $\mathsf{lag}(T, t_h + 1)$ could be as high as $(q+1) \cdot wt(T)$, if the subtask T_i of T scheduled at t_h is an MI , *i.e.,* is in A_q^0. Therefore, if a_q^0 is large, then LAG at $t_h + 1$ could exceed $qM + 1$. However, as we show below, if the number of MIs and SMIs scheduled at t_h is large, then the number of tasks that are inactive at t_h is also large, which can in turn be used to show that LAG does not increase across t_h. Specifically, we show that if $a_q^0 + a_q^1 > (q+1)(h-1)$, then $\mathsf{LAG}(\tau, t_h + 1) \leq \mathsf{LAG}(\tau, t_h) < qM + 1$, contradicting (T1). (Otherwise, the number of MIs and SMIs is not large enough for LAG to equal or exceed $qM + 1$.)

We begin by giving a lemma concerning the sum of the weights of tasks in I.

Lemma 21. *If* $\mathsf{LAG}(\tau, t_h + 1) > \mathsf{LAG}(\tau, t_h)$*, then* $\sum_{V \in I} wt(V) < h$.

Proof: By (17),

$$\begin{aligned}\mathsf{LAG}(\tau, t_h + 1) &= \mathsf{LAG}(\tau, t_h) + \sum_{T \in \tau} (\mathsf{A}(\mathsf{ideal}, T, t_h) - \mathcal{S}(T, t_h)) \\ &= \mathsf{LAG}(\tau, t_h) + \sum_{T \in A \cup B} (\mathsf{A}(\mathsf{ideal}, T, t_h)) - (M - h) \\ &\qquad \text{(by (23) and } \mathsf{A}(\mathsf{ideal}, T, t_h) = 0 \text{ for } T \text{ in } I\text{, and (32))} \\ &\leq \mathsf{LAG}(\tau, t_h) + \sum_{T \in A \cup B} wt(T) - (M - h) \qquad \text{(by (7)).}\end{aligned}$$

If $\mathsf{LAG}(\tau, t_h + 1) > \mathsf{LAG}(\tau, t_h)$, then by the derivation above,

$$\sum_{T \in A \cup B} wt(T) > M - h. \tag{33}$$

By (5), (22), and (23), $\sum_{T \in I} wt(T) \leq M - \sum_{T \in A \cup B} wt(T)$, which by (33) implies that $\sum_{T \in I} wt(T) < h$. ■

We next determine the largest number of MIs and SMIs that may be scheduled at t_h, for $\sum_{T \in I} wt(T) < h$ to hold. We begin with a lemma that gives the latest time that a subtask of a task in B may be scheduled if $a_q^0 > 0$ or ($a_q^1 > 0$ and $a_{q-1}^0 > 0$).

Lemma 22. *If $a_q^0 > 0$ (that is, an MI is scheduled at t_h), or ($a_q^1 > 0$ and $a_{q-1}^0 > 0$) (that is, an SMI, and a c-MI, where $0 < c < q$, is scheduled at t_h), then subtask U_j defined by Definition 8 is scheduled no later than $t_h - (q+2)$,* i.e., *$t_b \leq t_h - (q+2)$.*

Proof: If $a_q^0 > 0$ holds, then this lemma is immediate from Definitions 8, 7, and Lemma 19(b). (Note that Definitions 8 and 7 imply that U_j is scheduled before t_h.)

If $a_{q-1}^0 > 0$ holds, then a c-MI, where $0 < c < q$, say T_i, is scheduled at t_h. Hence, $\mathsf{d}(T_i) = t_h + 1 - c \leq t_h$ holds. By the definition of c-MI, the predecessor of T_i is not scheduled at $t_h - 1$. Hence, the deadline of every subtask scheduled at $t_h - 1$ is at most t_h. By Definition 2, $\mathsf{d}(U_j) \geq t_h + 1$. Therefore, U_j is not scheduled at $t_h - 1$.

If $a_q^1 > 0$ holds, then an SMI is scheduled at t_h, and its predecessor, which is an MI, is scheduled at $t_h - 1$. Therefore, by Lemma 19(b), U_j is not scheduled in $[t_h - 1 - (q+1), t_h - 1) = [t_h - (q+2), t_h - 1)$.

Thus, if both $a_{q-1}^0 > 0$ and $a_q^1 > 0$ hold, U_j is not scheduled later than $t_h - (q+3)$. ■

The lemma that follows is used to identify tasks that are inactive at t_h.

Lemma 23. *Let T be a task that is not scheduled at t_h. If T is scheduled in any of the slots in $[t_b + 1, t_h)$, then T is in I.*

Proof: T clearly is not in A. Because T is scheduled in $[t_b + 1, t_h)$, T is also not in B, by Definiton 7. ■

In the rest of this subsection, we let s denote the number of slots in $[t_b + 1, t_h)$. That is,

$$s \stackrel{\text{def}}{=} t_h - t_b - 1 \geq q + 1 \qquad \text{(by Lemma 22).} \tag{34}$$

We now determine a lower bound on the number of subtasks of tasks in I that may be scheduled in $[t_b + 1, t_h)$ as a function of a_q^0, a_q^1, h, and s. For this purpose, we assign subtasks scheduled in $[t_b + 1, t_h)$

to processors in a systematic way. This assignment is only for accounting purposes; subtasks need not be bound to processors in the actual schedule.

Processor groups. The assignment of subtasks to processors is based on the tasks scheduled at t_h. The M processors are partitioned into four disjoint sets, P_1, P_2, P_3, and P_4, based on the tasks scheduled at t_h, as follows.

$\boldsymbol{P_1}$ **:** By Lemma 18, there is at least one subtask W_ℓ scheduled at t_h such that $\mathsf{e}(W_\ell) \leq t_b$ and $\mathcal{S}(W,t) = 0$, for t in $[t_b, t_h)$. We assign one such subtask to the lone processor in this group. Hence, $|P_1| = 1$.

$\boldsymbol{P_2}$ **:** The h processors that are idle at t_h comprise this group. Thus, $|P_2| = h$.

$\boldsymbol{P_3}$ **:** This group consists of the $a_q^0 + a_q^1$ processors on which the a_q^0 MIs and a_q^1 SMIs are scheduled. Because either $a_q^0 > 1$ or $a_q^1 > 1$ holds, $|P_3| \geq 1$. τ^3 denotes the subset of all tasks scheduled on processors in P_3 at t_h.

$\boldsymbol{P_4}$ **:** Processors not assigned to P_1, P_2, or P_3 belong to this group. τ^4 denotes the subset of all tasks scheduled on processors in P_4 at t_h.

Subtask assignment in $\boldsymbol{[t_b+1, t_h)}$. Subtasks scheduled in $[t_b + 1, t_h)$ are assigned to processors by the following rules. Tasks in τ^3 and τ^4 are assigned to the same processor that they are assigned to in t_h, in every slot in which they are scheduled in $[t_b + 1, t_h)$. (It is trivial that such an assignment is possible since by the processor groups defined above, $|\tau^3| + |\tau^4| = P_3 + P_4 \leq M - h - 1 < M$.) Subtasks of tasks not in τ^3 or τ^4 may be assigned to any processor.

The next three lemmas bound the number of subtasks of tasks in I scheduled in $[t_b + 1, t_h)$. These lemmas assume that the assignment of subtasks to processors in $[t_b + 1, t_h)$ follows the rules described above. In these lemmas we assume that either $a_q^0 \geq 1$ or ($a_q^1 \geq 1$ and $a_{q-1}^0 \geq 1$) holds.

Lemma 24. *The number of subtasks of tasks in I that are scheduled in $[t_b + 1, t_h)$ is at least $s \cdot (h + 1) + (a_q^0 + a_q^1)$.*

Proof: We first make the following two claims.

> **Claim 2.** *Let T_i be a subtask assigned to a processor in P_1 or P_2 in $[t_b + 1, t_h)$. Then, T is in I.*
> **Proof:** By our assignment of subtasks to processors, tasks assigned to processors in P_1 or P_2 in $[t_b + 1, t_h)$ are not scheduled at t_h. Therefore, T is not scheduled at t_h. Hence, by Lemma 23, T is inactive at t_h, *i.e.*, is in I. ■
>
> **Claim 3.** *At least one of the subtasks assigned to each processor in P_3 in $[t_b + 1, t_h)$ is a subtask of a task in I.*
>
> **Proof:** Let P_3^x be any processor in P_3, and let T_i be the subtask scheduled on P_3^x at t_h. Then, T_i is either an MI or an SMI. In the former case, by the definition of an MI, $\mathcal{S}(T, t_h - 1) = 0$, and in the latter, by the definition of an SMI, $\mathcal{S}(T, t_h - 2) = 0$. By Lemma 22, $t_b \leq t_h - (q + 2)$. Thus, since $q \geq 1$, and by Lemma 18, there is no hole in any slot in $[t_b, t_h)$, there is no hole in slot $t_h - 2$ or $t_h - 1$. Thus, a subtask of a task V other than T is assigned to P_3^x in one of these two slots. By our subtask assignment, V is not scheduled at t_h; thus, by Lemma 23, $V \in I$. ■

The lemma follows from the definition of s in (34), and Claims 2 and 3 above. ■

Lemma 25. *The sum of the weights of the tasks in I is at least $\frac{(h+1)\cdot s}{s+q+1} + \frac{a_q^0+a_q^1}{s+q+1}$.*

Proof: Let V_k be a subtask of a task V in I that is scheduled in $[t_b+1, t_h)$. Then, by Definition 1, $\mathsf{d}(V_k) \le t_h$. By Definition 8, U_j is scheduled at t_b, and by Definition 2, $\mathsf{d}(U_j) \ge t_h + 1$. Because V_k with an earlier deadline than U_j is scheduled later than t_b, either $\mathsf{r}(V_k) \ge t_b + 1$ or V_k's predecessor V_j, where $j < k$, is scheduled at t_b. In the latter case, by Lemma 7(b), $tardiness(V_j) \le q$, and hence, $\mathsf{d}(V_j) \ge t_b - q + 1$, which, by Lemma 2, implies $\mathsf{r}(V_k) \ge t_b - q$. Thus, we have the following.

$$(\forall V_k : V \in I :: ((u \in [t_b+1, t_h) \wedge \mathcal{S}(V_k, u) = 1) \Rightarrow (\mathsf{r}(V_k) \ge t_b - q \wedge \mathsf{d}(V_k) \le t_h))) \tag{35}$$

We next show that $wt(V) \ge \frac{V.n}{s+q+1}$, where $V.n$ is the number of subtasks of V scheduled in $[t_b+1, t_h)$. Let V_k and V_ℓ denote the first and final subtasks of V scheduled in $[t_b+1, t_h)$. Then, by (35), $\mathsf{r}(V_k) \ge t_b - q$ and $\mathsf{d}(V_\ell) \le t_h$. Hence,

$$\mathsf{d}(V_\ell) - \mathsf{r}(V_k) \le t_h - t_b + q = s + q + 1 \quad \text{(by the definition of } s \text{ in (34)).} \tag{36}$$

By (1),

$$\begin{aligned} \mathsf{d}(V_\ell) - \mathsf{r}(V_k) &= \left\lceil \frac{\ell}{wt(V)} \right\rceil - \left\lfloor \frac{k-1}{wt(V)} \right\rfloor + \Theta(V_\ell) - \Theta(V_k) \\ &\ge \left\lceil \frac{\ell}{wt(V)} \right\rceil - \left\lfloor \frac{k-1}{wt(V)} \right\rfloor \qquad \text{(by } \ell > k \text{ and (2)).} \end{aligned} \tag{37}$$

By (36) and (37), we have $\left\lceil \frac{\ell}{wt(V)} \right\rceil - \left\lfloor \frac{k-1}{wt(V)} \right\rfloor \le s+q+1$, which implies $\frac{\ell}{wt(V)} - \frac{k-1}{wt(V)} \le s+q+1$, *i.e.*,

$$wt(V) \ge \frac{\ell - k + 1}{s+q+1} \ge \frac{V.n}{s+q+1} \qquad \begin{array}{l}\text{(because } V.n = \ell - k + 1 \text{ if } V \text{ is periodic and} \\ V.n \le \ell - k + 1 \text{ if } V \text{ is IS or GIS)}\end{array}.$$

Therefore, we have $\sum_{V\in I} wt(V) \ge \sum_{V \in I} \frac{V.n}{s+q+1} \ge \frac{(h+1)\cdot s}{s+q+1} + \frac{a_1^0+a_1^1}{s+q+1}$, where the last inequality is by Lemma 24. ■

Lemma 26. *If $\mathsf{LAG}(\tau, t_h+1) > \mathsf{LAG}(\tau, t_h)$ and either $a_q^0 \ge 1$ or ($a_q^1 \ge 1$ and $a_{q-1}^0 \ge 1$), then $a_q^0 + a_q^1 \le \min((h-1)(q+1)-1, M-h-1)$.*

Proof: By Lemma 21, if $\mathsf{LAG}(\tau, t_h+1) > \mathsf{LAG}(\tau, t_h)$, then $\sum_{V \in I} wt(V) < h$. By Lemma 25, $\frac{(h+1)\cdot s}{s+q+1} + \frac{a_q^0+a_q^1}{s+q+1} \le \sum_{V\in I} wt(V)$. Therefore, $\frac{(h+1)\cdot s}{s+q+1} + \frac{a_q^0+a_q^1}{s+q+1} < h$, which implies that

$$\begin{aligned} a_q^0 + a_q^1 &< h(q+1) - s \\ &\le h(q+1) - (q+1) \qquad \text{(by (34))} \\ &= (h-1)(q+1). \end{aligned} \tag{38}$$

Also, there are h holes in t_h, and by Lemma 18, $a_0 \ge 1$. Therefore, by (32),

$$a_q^0 + a_q^1 \le M - h - 1. \tag{39}$$

(38) and (39) imply that $a_q^0 + a_q^1 \le \min((h-1)(q+1)-1, M-h-1)$. ■

We now conclude Case B by establishing the following.

Lemma 27. *If* $a_q^0 > 0$ *or* ($a_q^1 > 0$ *and* $a_{q-1}^0 > 0$)*, then* $\mathsf{LAG}(\tau, t_h + 1) < qM + 1$.

Proof: Because $W_{\max} < 1$, assuming $W_{\max} \geq 2/3$ (because, as discussed earlier, a_0, a_{q-1}, and a_q^i are not dependent on $W_{\max}$), by (31), we have

$$\begin{aligned}\mathsf{LAG}(\tau, t_h + 1) \quad &< \quad a_0 \cdot W_{\max} + ((q+1) \cdot W_{\max}) \cdot (a_q^0 + a_q^1) \\ &\quad\quad + (a_{q-1} + a_q^2) \cdot ((q+3) \cdot W_{\max} - 2). \end{aligned} \tag{40}$$

By Lemma 26, if $\mathsf{LAG}(\tau, t_h + 1) > \mathsf{LAG}(\tau, t_h)$, then $a_q^0 + a_q^1 \leq \min((h-1)(q+1) - 1, M - h - 1)$. By Lemmas 12(a)–(g) (and as can be seen from the coefficients of the a_i terms in (40)), the lag bounds for tasks in $A_q^0 \cup\ A_q^1$ are higher than those for the other tasks. Hence, $\mathsf{LAG}(\tau, t_h + 1)$ is maximized when $a_q^0 + a_q^1 = \min((h-1)(q+1) - 1, M - h - 1)$. We assume this is the case. Note that

$$\min((h-1)(q+1) - 1, M - h - 1) = \begin{cases} (h-1)(q+1) - 1, & h \leq \frac{M+1+q}{q+2} \\ M - h - 1, & \text{otherwise.} \end{cases} \tag{41}$$

Based on (41), we consider two cases.

Case 1: $h > \frac{M+1+q}{q+2}$. For this case, LAG is maximized when $a_q^0 + a_q^1 = M - h - 1$, and hence, by (32), $a_0 + a_{q-1} + a_q^2 = M - h - (a_1^0 + a_1^1) = 1$. Because, by Lemma 18, $a_0 > 0$, we have $a_0 = 1$, and hence, $a_{q-1} = a_q^2 = 0$. Substituting $a_0 = 1$, $a_q^2 = a_{q-1} = 0$, and $a_q^0 + a_q^1 = M - h - 1$ in (40), we have $\mathsf{LAG}(\tau, t_h + 1) < W_{\max} + (q+1) \cdot W_{\max} \cdot (a_q^0 + a_q^1) = W_{\max} + (q+1) \cdot W_{\max} \cdot (M - h - 1) < W_{\max} + (q+1) \cdot W_{\max} \cdot \left(M - \frac{M+q+1}{q+2} - 1\right)$ (where the last inequality is by the condition of Case 1, namely, $h > \frac{M+1+q}{q+2}$). If $qM + 1 \leq \mathsf{LAG}(\tau, t_h + 1)$, then $W_{\max} + (q+1) \cdot W_{\max} \cdot \left(M - \frac{M+q+1}{q+2} - 1\right) > qM + 1$, which implies that $W_{\max} > \frac{Mq(q+2)+q+2}{M(q+1)^2 - (2q^2+4q+1)}$, which is greater than $\frac{q+2}{q+3}$ for all $q \geq 1$ and $M \geq 2$. This contradicts (W), and hence, $\mathsf{LAG}(\tau, t_h + 1) < qM + 1$.

Case 2: $h \leq \frac{M+1+q}{q+2}$. For this case, LAG is maximized when $a_q^0 + a_q^1 = (h-1)(q+1) - 1$. By (32), we have $a_{q-1} + a_q^2 = M - h - (a_0 + a_q^0 + a_q^1) = M - h - a_0 - (hq + h - q - 2)$. Therefore, by (40),

$$\begin{aligned} &\mathsf{LAG}(\tau, t_h + 1) \\ &\quad < \quad a_0 \cdot W_{\max} + (q+1) \cdot W_{\max} \cdot (a_q^0 + a_q^1) + ((q+3) \cdot W_{\max} - 2) \cdot (a_{q-1} + a_q^2) \\ &\quad = \quad a_0 \cdot W_{\max} + (q+1) \cdot W_{\max} \cdot (hq + h - q - 2) \\ &\quad\quad\quad + ((q+3) \cdot W_{\max} - 2)(M - 2h - a_0 - hq + q + 2). \end{aligned} \tag{42}$$

If $qM + 1 \leq \mathsf{LAG}(\tau, t_h + 1)$, then the expression on the right-hand side of (42) exceeds $qM + 1$, which implies that $W_{\max} > \frac{(q+2)M + 2q + 5 - 4h - 2a_0 - 2hq}{(q+3)M + 2q + 4 - 5h - (2+q)a_0 - 3hq}$. Let $f \stackrel{\text{def}}{=} \frac{(q+2)M + 2q + 5 - 4h - 2a_0 - 2hq}{(q+3)M + 2q + 4 - 5h - (2+q)a_0 - 3hq}$, and let Y denote the denominator, $(q+3)M + 2q + 4 - 5h - (2+q)a_0 - 3hq$, of f. To show that the lemma holds for this case, we show that unless $W_{\max}$ exceeds $\frac{q+2}{q+3}$, $qM + 1 > \mathsf{LAG}(\tau, t_h + 1)$. For this purpose, we determine a lower bound to the value of f. Note that for a given number of processors, M, and tardiness, q, f varies with a_0 and h. Because $a_q^0 + a_q^1 = (h-1)(q+1) - 1 > 0$, we have $h > \frac{q+2}{q+1}$; hence, because h is integral, $h \geq 2$ holds. The first derivative of f with respect to h is $\frac{M(q^2+q-2) + a_0(2q^2+2q-2) + 2q^2+9q+9}{Y^2}$, which is non-negative for all $a_0 \geq 0$, and that with respect to a_0 is $\frac{M(q^2+2q-2) + h(2-2q-2q^2) + 2q^2+5q+2}{Y^2}$, which is non-negative for $h \leq \frac{M(q^2+2q-2)+2q^2+5q+2}{2q^2+2q-2}$. Thus, f is minimized when $h = 2$, and because $\frac{M(q^2+2q-2)+2q^2+5q+2}{2q^2+2q-2} \geq \frac{M+1+q}{q+2}$ (where $\frac{M+1+q}{q+2} \geq h$ holds for this case), when $a_0 = 1$. When $h = 2$ and $a_0 = 1$ hold, $f = \frac{qM+2M-2q-5}{qM+3M-5q-8} > \frac{q+2}{q+3}$, for all M (since when $h = 2$ and $a_0 = 1$, we have $M \geq 4$). Hence, $W_{\max} > \frac{q+2}{q+3}$, which is a violation of (W), and the lemma follows for this case. ■

Thus, if an MI or an SMI and a c-MI are scheduled in t_h, then (T1) is contradicted.

3.8. Case C: ($A^0_q = \emptyset$ *and* $A^1_q \neq \emptyset$ *and* $A^0_{q-1} = \emptyset$)

For this case, we show that if $\mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) > qM + 1$, then there exists another concrete task system τ', obtained from τ by removing certain subtasks, such that LAG of τ' at $t_h - 1$ in an EPDF schedule $\mathcal{S}'$ is greater than $qM + 1$ contradicting the minimality of t_h (in Definition 5). Our approach is to identify task subsets, determine the lag for tasks in each subset in $\mathcal{S}'$ at $t_h - 1$, and use task lags to determine the LAG of τ' at $t_h - 1$. We begin by defining needed subsets of subtasks and tasks.

In this case, since no MI is scheduled in slot t_h, t_b (in Definition 7) can be as late as $t_h - 1$. This is stated below.

$$t_b \leq t_h - 1 \tag{43}$$

Let t'_b be defined as follows.

Definition 9:. t'_b denotes the latest time, if any, before $t_h - 1$ that a subtask with deadline at or after t_h is scheduled.

Since at least one SMI is scheduled at t_h, at least one MI is scheduled at $t_h - 1$. Therefore, by Lemma 19(a), the following holds.

> **(C)** The deadline of every subtask scheduled in any slot in $[t_h - (q+2), t_h - 1)$ is at or before $t_h - q$.

Since $q \geq 1$ holds, (C) implies the following.

$$\text{when it exists, } t'_b \leq t_h - (q+3) \tag{44}$$

Let τ^1_s through τ^8_s be subsets of subtasks defined as follows. In the definitions that follow, when we say that T_i is *ready* at t'_b, we mean that $\mathsf{e}(T_i) \leq t'_b$, and T_i's predecessor, if any, is scheduled before t'_b.

$\tau^1_s \stackrel{\text{def}}{=} \{T_i \mid T_i$ is either the critical subtask at t_h of a task in $B(t_h)$ or the critical subtask at $t_h - 1$ of a task in $B(t_h - 1)$, t'_b exists, T_i is scheduled at or before t'_b, and T is not scheduled at $t_h\}$

$\tau^2_s \stackrel{\text{def}}{=} \{T_i \mid \mathsf{d}(T_i) \geq t_h$, T_i is scheduled at $t_h - 1$, and T is not scheduled at $t_h\}$

$\tau^3_s \stackrel{\text{def}}{=} \{T_i \mid T \in A_0(t_h)$, T_i is scheduled at t_h, and T_i is ready at or before $t_h - (q+3)$ in $\mathcal{S}\}$

$\tau^4_s \stackrel{\text{def}}{=} \{T_i \mid T \in A_0(t_h)$, T_i is scheduled at t_h, and T_i is not ready at or before $t_h - (q+3)$ in $\mathcal{S}\}$

$\tau^5_s \stackrel{\text{def}}{=} \{T_i \mid T \in (A^1_q(t_h) \cup A^2_q(t_h) \cup A_{q-1}(t_h))$, T_i is scheduled at t_h, and T is scheduled at $t_h - 1\}$

$\tau^6_s \stackrel{\text{def}}{=} \{T_i \mid T_i$ is scheduled at $t_h - 1$, $T_i \notin \tau^2_s$ (*i.e.*, $\mathsf{d}(T_i) < t_h$), and T is not scheduled at $t_h\}$

$\tau^7_s \stackrel{\text{def}}{=} \{T_i \mid T_i$ is the predecessor of a subtask in τ^1_s and $\mathsf{d}(T_i) = t_h\}$

$\tau^8_s \stackrel{\text{def}}{=} \{T_i \mid T_i$ is the predecessor of a subtask in τ^2_s and $\mathsf{d}(T_i) = t_h\}$

Let τ^i denote the set of all tasks with a subtask in τ_s^i, for all $1 \leq i \leq 8$. Note that $\tau^7 \subseteq \tau^1$ and $\tau^8 \subseteq \tau^2$ hold.

The following lemma establishes some properties concerning the subsets of subtasks and tasks defined above. It is proved in an appendix.

Lemma 28. *The following properties hold for subsets τ_s^i and τ^i defined above, where $1 \leq i \leq 8$.*

(a) *For every task T, there is at most one subtask in $(\tau_s^1 \cup \tau_s^2 \cup \tau_s^6)$.*

(b) *Let T_i scheduled at t_h be the subtask of a task T in $A_q(t_h)$ or $A_{q-1}(t_h)$. Then, T_i is in τ_s^5.*

(c) $\tau^7 \subseteq \tau^1$ *and* $\tau^8 \subseteq \tau^2$.

(d) *Subsets τ^i, where $1 \leq i \leq 6$, are pairwise disjoint.*

Let

$$\tau_s^R \stackrel{\text{def}}{=} \tau_s^1 \cup \tau_s^2 \cup \tau_s^3 \cup \tau_s^7 \cup \tau_s^8, \tag{45}$$

and let τ' be a concrete GIS task system obtained from τ by removing all the subtasks in τ_s^R. Let $\mathcal{S}'$ be an EPDF schedule for τ' such that ties among subtasks with equal deadlines are resolved in the same way as they are resolved in $\mathcal{S}$. Our goal is to show that $\mathsf{LAG}(\tau', t_h - 1, \mathcal{S}') \geq qM + 1$, and derive a contradiction to the minimality of t_h in Definition 5. For this purpose, in the next few lemmas (proved in an appendix), we establish lag bounds in $\mathcal{S}'$ for tasks with subtasks in the subsets defined above. We will denote the ideal schedule for τ as ideal_τ and that for τ' as $\mathsf{ideal}_{\tau'}$.

Lemma 29. *Let T be a task with a subtask in τ_s^1 or τ_s^2. Then,* $\mathsf{lag}(T, t_h - 1, \mathcal{S}') = \mathsf{lag}(T, t_h + 1, \mathcal{S})$.

Lemma 30. *Let T be a task with a subtask in τ_s^3. Then,* $\mathsf{lag}(T, t_h - 1, \mathcal{S}') > \mathsf{lag}(T, t_h + 1, \mathcal{S}) - 1/(q + 2)$.

Lemma 31. *Let T be a task with a subtask in τ_s^4. Then,* $\mathsf{lag}(T, t_h - 1, \mathcal{S}') \geq \mathsf{lag}(T, t_h + 1, \mathcal{S}) - 2 \cdot W_{\max} + 1$.

Lemma 32. *Let T be a task with a subtask in τ_s^5. Then,* $\mathsf{lag}(T, t_h - 1, \mathcal{S}') \geq \mathsf{lag}(T, t_h + 1, \mathcal{S}) + 2 - 2 \cdot W_{\max}$.

Lemma 33. *Let T be a task with a subtask in τ_s^6. Then,* $\mathsf{lag}(T, t_h - 1, \mathcal{S}') > \mathsf{lag}(T, t_h + 1, \mathcal{S})$.

Let $\tau^c = \tau' \setminus (\cup_{i=1}^6 \tau_i)$. Because τ and τ' are concrete instantiations of the same non-concrete task system, they both contain the same tasks, and hence, $\tau^c = \tau \setminus (\cup_{i=1}^6 \tau_i)$. We show the following concerning the lag of a task in τ^c at $t_h - 1$ in $\mathcal{S}'$. (This lemma is also proved in an appendix.)

Lemma 34. *Let T be a task in τ^c. Then,* $\mathsf{lag}(T, t_h - 1, \mathcal{S}') = \mathsf{lag}(T, t_h + 1, \mathcal{S})$.

Having determined bounds for the lags of tasks at $t_h - 1$ in $\mathcal{S}'$, we now determine a lower bound for the LAG of τ' at $t_h - 1$ in $\mathcal{S}'$, and show that if (W) holds, then $\mathsf{LAG}(\tau', t_h - 1, \mathcal{S}') \geq qM + 1$.

Lemma 35. *If either $(W_{\max} \leq \frac{q+3}{2q+4}$ and $a_0 \leq \frac{(M-h)\cdot(q+1)}{q+2})$ or $(W_{\max} > \frac{q+3}{2q+4}$ and $a_0 \leq 2(M - h)(1 - W_{\max}))$, then* $\mathsf{LAG}(\tau', t_h - 1, \mathcal{S}') \geq qM + 1$.

Proof: By (16),

$$
\begin{aligned}
&\mathsf{LAG}(\tau', t_h - 1, \mathcal{S}') \\
&= \sum_{T \in \tau'} \mathsf{lag}(T, t_h - 1, \mathcal{S}') \\
&= \sum_{T \in \tau} \mathsf{lag}(T, t_h - 1, \mathcal{S}') \qquad \text{(by the construction of } \tau'\text{)} \\
&= \sum_{i=1}^{6} \sum_{T \in \tau^i} \mathsf{lag}(T, t_h - 1, \mathcal{S}') + \sum_{T \in \tau^c} \mathsf{lag}(T, t_h - 1, \mathcal{S}') \\
&\qquad \text{(by Lemmas 28(c) and (d), and because } \tau^c = \tau \setminus \cup_{i=1}^{6} \tau^i\text{)} \\
&\geq \sum_{T \in \tau^1 \cup \tau^2 \cup \tau^6 \cup \tau^c} \mathsf{lag}(T, t_h + 1, \mathcal{S}) + \sum_{i=3}^{5} \sum_{T \in \tau^i} \mathsf{lag}(T, t_h - 1, \mathcal{S}') \\
&\qquad \text{(by Lemmas 29, 33, and 34)} \\
&\geq \sum_{i=1}^{6} \sum_{T \in \tau^i} \mathsf{lag}(T, t_h + 1, \mathcal{S}) + \sum_{T \in \tau^c} \mathsf{lag}(T, t_h + 1, \mathcal{S}) - |\tau^3| \cdot \frac{1}{q+2} \\
&\quad + |\tau^4| \cdot (1 - 2W_{\max}) + |\tau^5| \cdot (2 - 2W_{\max}) \qquad \text{(by Lemmas 30–32)} \\
&= \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) - |\tau^3| \cdot \frac{1}{q+2} - |\tau^4| \cdot (2W_{\max} - 1) + |\tau^5| \cdot (2 - 2W_{\max}) \qquad (46) \\
&\qquad \text{(by the definitions of sets } \tau^i\text{, where } 1 \leq i \leq 6\text{, and } \tau^c\text{).}
\end{aligned}
$$

Note that

$$|\tau^3| + |\tau^4| = a_0. \qquad (47)$$

By Lemma 28(b), $|\tau^5| = |A_q| + |A_{q-1}| = a_q + a_{q-1}$. By the definitions of A_q, A_q^0, A_q^1, and A_q^2, and by (25)–(27), $a_q = a_q^0 + a_q^1 + a_q^2$. However, because no MI is scheduled at t_h by the conditions of Case C, $a_q^0 = 0$, and hence,

$$|\tau^5| = a_q^1 + a_q^2 + a_{q-1} = M - h - a_0 \qquad \text{(by (32)).} \qquad (48)$$

We now consider the following two cases based on the statement of the lemma.

Case 1: $W_{\max} > \frac{q+3}{2q+4}$ and $a_0 \leq 2(M-h)(1-W_{\max})$. Since $W_{\max} > \frac{q+3}{2q+4}$, $2W_{\max} - 1 > \frac{1}{q+2}$ holds. By (46),

$$
\begin{aligned}
&\mathsf{LAG}(\tau', t_h - 1, \mathcal{S}') \\
&\geq \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) - |\tau^3| \cdot \frac{1}{q+2} - |\tau^4| \cdot (2W_{\max} - 1) + |\tau^5| \cdot (2 - 2W_{\max}) \\
&\geq \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) - |\tau^3| \cdot (2W_{\max} - 1) - |\tau^4| \cdot (2W_{\max} - 1) + |\tau^5| \cdot (2 - 2W_{\max}) \\
&\qquad \text{(because as mentioned above, } 2W_{\max} - 1 > \tfrac{1}{q+2}\text{)} \\
&= \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) - a_0 \cdot (2W_{\max} - 1) + (M - h - a_0) \cdot (2 - 2W_{\max}) \\
&\qquad \text{(by (47) and (48))} \\
&= \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) - a_0 + (M - h) \cdot (2 - 2W_{\max}) \\
&\geq \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) \qquad \text{(because } 2(M-h) \cdot (1 - W_{\max}) \geq a_0 \text{ for this case)} \\
&\geq qM + 1 \qquad \text{(by (T1)).} \qquad (49)
\end{aligned}
$$

Case 2: $W_{\max} \leq \frac{q+3}{2q+4}$ and $a_0 \leq \frac{(M-h)\cdot(q+1)}{q+2}$. Since $W_{\max} \leq \frac{q+3}{2q+4}$, $2 \cdot W_{\max} - 1 \leq \frac{1}{q+2}$ holds. As with Case 1, by (46),

$$
\begin{aligned}
&\mathsf{LAG}(\tau', t_h - 1, \mathcal{S}') \\
&\geq \quad \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) - |\tau^3| \cdot \frac{1}{q+2} - |\tau^4| \cdot (2W_{\max} - 1) + |\tau^5| \cdot (2 - 2 \cdot W_{\max}) \\
&\geq \quad \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) - |\tau^3| \cdot \frac{1}{q+2} - |\tau^4| \cdot \frac{1}{q+2} + |\tau^5| \cdot (2 - 2 \cdot W_{\max}) \\
&\qquad\qquad (\text{because } 2 \cdot W_{\max} - 1 \leq \tfrac{1}{q+2}) \\
&\geq \quad \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) - |\tau^3| \cdot \frac{1}{q+2} - |\tau^4| \cdot \frac{1}{q+2} + |\tau^5| \cdot \frac{2q+2}{2(q+2)} \\
&\qquad\qquad (\text{because } W_{\max} \leq \tfrac{q+3}{2(q+2)}) \\
&= \quad \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) - a_0 \cdot \frac{1}{q+2} + (M - h - a_0) \cdot \frac{q+1}{q+2} \\
&\qquad\qquad (\text{by (47) and (48)}) \\
&= \quad \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) - a_0 + (M - h) \cdot \frac{q+1}{q+2} \\
&\geq \quad \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) \ (\text{because } a_0 \leq \tfrac{(M-h)\cdot(q+1)}{q+2} \text{ for this case}) \\
&\geq \quad qM + 1 \qquad\qquad (\text{by (T1)}). \qquad\qquad (50)
\end{aligned}
$$

The lemma follows from (49) and (50), and by the conditions of Cases 1 and 2, respectively. ■

In completing Case C, we make use of this auxiliary algebraic lemma, proved in an appendix.

Lemma 36. *The roots of* $f(W_{\max}) \stackrel{\text{def}}{=} 2(M-h)(q+1)W_{\max}^2 - (q+2)(M-h)W_{\max} - ((q-1)M+1+h) = 0$ *are* $W_{\max} = \frac{(q+2)(M-h)\pm\sqrt{9q^2(M-h)^2+\Delta}}{4(M-h)(q+1)}$, *where* $\Delta = 4(M-h)(M(q-1) + h(2q^2+q+1) + 2q + 2)$.

We conclude this case by establishing the following lemma.

Lemma 37. *If either* ($W_{\max} \leq \frac{q+3}{2q+4}$ *and* $a_0 > \frac{(M-h)\cdot(q+1)}{q+2}$) *or* ($W_{\max} > \frac{q+3}{2q+4}$ *and* $a_0 > 2(M-h)(1 - W_{\max})$), *then* $\mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) < qM + 1$.

Proof: We consider two cases based on the statement of the lemma.

Case 1: $W_{\max} > \frac{q+3}{2q+4}$ and $a_0 > 2(M-h)(1 - W_{\max})$. By (30),

$$
\begin{aligned}
&\mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) \\
&< \quad a_0 \cdot W_{\max} + a_q^0(q+1)W_{\max} + a_{q-1} \cdot q \cdot W_{\max} + a_q^1((q+2)W_{\max} - 1) \\
&\qquad + a_q^2((q+3)W_{\max} - 2) \\
&< \quad a_0 \cdot W_{\max} + a_q^0(q+1)W_{\max} + (a_{q-1} + a_q^1)((q+2)W_{\max} - 1) \\
&\qquad + a_q^2((q+3)W_{\max} - 2) \\
&\qquad (\text{by the conditions of Case 1, } W_{\max} > \tfrac{q+3}{2q+4} \geq \tfrac{1}{2}; \text{ thus,} \\
&\qquad q \cdot W_{\max} < (q+2)W_{\max} - 1 \text{ holds}) \\
&< \quad a_0 \cdot W_{\max} + a_q^0(q+1)W_{\max} + (a_{q-1} + a_q^1 + a_q^2)((q+2)W_{\max} - 1) \\
&\qquad (\text{because } W_{\max} < 1)
\end{aligned}
$$

$$
\begin{aligned}
&\leq \quad a_0 \cdot W_{\max} + (M-h-a_0)\cdot((q+2)W_{\max}-1) \\
&\qquad \text{(by (32) because } a_q^0 = 0 \text{ by the conditions of Case C)} \\
&= \quad a_0 \cdot (1-(q+1)W_{\max}) + (M-h)\cdot((q+2)W_{\max}-1) \\
&< \quad 2(M-h)(1-W_{\max})\cdot(1-(q+1)W_{\max}) + (M-h)\cdot((q+2)W_{\max}-1) \\
&\qquad \text{(because } W_{\max} > \tfrac{q+3}{2q+4} \geq \tfrac{1}{q+1} \text{ for all } q \geq 1,\ 1-(q+1)W_{\max} < 0\text{; also, by the} \\
&\qquad \text{conditions of Case 1, } a_0 > 2(M-h)(1-W_{\max})) \\
&= \quad 2(M-h)(q+1)W_{\max}^2 - (q+2)(M-h)W_{\max} + M - h.
\end{aligned}
$$

We next show that $\mathsf{LAG}(\tau, t_h+1, \mathcal{S}) < qM+1$ holds (if (W) holds). Suppose to the contrary that $\mathsf{LAG}(\tau, t_h+1, \mathcal{S}) \geq qM+1$; then by the derivation above

$$2(M-h)(q+1)W_{\max}^2 - (q+2)(M-h)W_{\max} - ((q-1)M+1+h) > 0. \tag{51}$$

By Lemma 36, the roots of $f(W_{\max}) = 2(M-h)(q+1)W_{\max}^2 - (q+2)(M-h)W_{\max} - ((q-1)M+1+h) = 0$ are $W_{\max} = \frac{(q+2)(M-h)\pm\sqrt{9q^2(M-h)^2+\Delta}}{4(M-h)(q+1)}$, where $\Delta = 4(M-h)(M(q-1)+h(2q^2+q+1)+2q+2)$. Let $W_{\max,1} = \frac{(q+2)(M-h)+\sqrt{9q^2(M-h)^2+\Delta}}{4(M-h)(q+1)}$ and $W_{\max,2} = \frac{(q+2)(M-h)-\sqrt{9q^2(M-h)^2+\Delta}}{4(M-h)(q+1)}$. Since $0 < h < M$ and $q \geq 1$ hold, $\Delta > 0$ holds, and hence, $\sqrt{9q^2(M-h)^2+\Delta}$ is greater than $3q(M-h)$. Note that $W_{\max,1} > \frac{(q+2)(M-h)+3(M-h)q}{4(M-h)(q+1)} = \frac{4q+2}{4q+4} > 0$. Also, because $h < M$, $3q(M-h) \geq (q+2)(M-h)$ for all $q \geq 1$. Therefore, $W_{\max,2} < 0$. The first derivative of $f(W_{\max})$ with respect to $W_{\max}$ is given by $f'(W_{\max}) = 4(M-h)(q+1)W_{\max} - (q+2)(M-h)$, which is positive for $W_{\max} > \frac{q+2}{4q+4}$. Hence, $f(W_{\max})$ is an increasing function of $W_{\max}$ for $W_{\max} \geq \frac{q+2}{4q+4}$; further, the following hold: $W_{\max,1} > \frac{4q+2}{4q+4} > \frac{q+2}{4q+4}$, $f(W_{\max,1}) = 0$, and $f(W_{\max})$ is quadratic. Therfore, we have $f(W_{\max}) < 0$ for $W_{\max,2} < 0 < W_{\max} < W_{\max,1}$. Because as mentioned earlier, $W_{\max,1} > \frac{(q+2)(M-h)+3(M-h)q}{4(M-h)(q+1)} = \frac{4q+2}{4q+4} > \frac{q+2}{q+3}$, it follows that, for all $0 < W_{\max} \leq \frac{q+2}{q+3}$, $f(W_{\max}) < 0$. By (W), $W_{\max} \leq \frac{q+2}{q+3}$ holds, and hence, (51) does not hold, implying that $\mathsf{LAG}(\tau, t_h+1) < qM+1$. Thus, by the conditions of Case 1, if $W_{\max} > \frac{q+3}{2q+4}$ and $a_0 > 2(M-h)(1-W_{\max})$, then $\mathsf{LAG}(\tau, t_h+1, \mathcal{S}) < qM+1$ follows.

Case 2: $\boldsymbol{W_{\max} \leq \frac{q+3}{2q+4}}$ and $\boldsymbol{a_0 > \frac{(M-h)\cdot(q+1)}{q+2}}$. Because $\frac{q+3}{2q+4} \leq \frac{2}{3}$, for all $q \geq 1$, $q \cdot W_{\max} \geq (q+3)W_{\max} - 2$ holds. Hence, by (30), we have

$$
\begin{aligned}
&\mathsf{LAG}(\tau, t_h+1, \mathcal{S}) \\
&\quad < \quad a_0 \cdot W_{\max} + (a_{q-1} + a_q^2)q \cdot W_{\max} + a_q^0(q+1)W_{\max} + a_q^1((q+2)W_{\max}-1) \\
&\quad = \quad a_0 \cdot W_{\max} + (a_{q-1} + a_q^2)q \cdot W_{\max} + a_q^1((q+2)W_{\max}-1) \\
&\qquad\qquad \text{(because } a_q^0 = 0 \text{ by the conditions of Case C).}
\end{aligned} \tag{52}
$$

We consider two subcases based on the value of $W_{\max}$.

Subcase 2(a): $\boldsymbol{\frac{1}{2} < W_{\max} \leq \frac{q+3}{2q+4}}$. For this case, $(q+2)W_{\max} - 1 > q \cdot W_{\max}$ holds. Hence, by (52), we have

$$
\begin{aligned}
&\mathsf{LAG}(\tau, t_h+1, \mathcal{S}) \\
&\quad < \quad a_0 \cdot W_{\max} + (a_{q-1} + a_q^2 + a_q^1)((q+2)W_{\max}-1) \\
&\quad \leq \quad a_0 \cdot W_{\max} + (M-h-a_0)\cdot((q+2)W_{\max}-1) \ \ \text{(by (32) because } a_q^0 = 0) \\
&\quad = \quad a_0 \cdot (1-(q+1)W_{\max}) + (M-h)\cdot((q+2)W_{\max}-1).
\end{aligned} \tag{53}
$$

Let $f(a_0, W_{\max}) \stackrel{\text{def}}{=} a_0 \cdot (1-(q+1)W_{\max}) + (M-h)\cdot((q+2)W_{\max}-1)$, the right-hand side of the above inequality. Our goal is to determine an upper bound for $f(a_0, W_{\max})$. We first show that $f(a_0, W_{\max})$ is an increasing function of $W_{\max}$ for all $a_0 \geq 0$, and a decreasing function of a_0, for any $W_{\max} \geq \frac{1}{q+1}$. (In the description that follows, we assume a_0 and $W_{\max}$ are non-negative.) The first derivative of $f(a_0, W_{\max})$ with respect to $W_{\max}$ is $(M-h)(q+2) - a_0(q+1)$. Therefore, since $a_0 \leq M-h$ and $M-h > 0$, it follows that $(M-h)(q+2) - a_0(q+1)$ is positive for all $q \geq 0$. Hence, $f(a_0, W_{\max})$ is an increasing function of $W_{\max}$ for all valid a_0. Further, $f(a_0, W_{\max})$ is a non-decreasing function of a_0 for all $W_{\max} \leq \frac{1}{q+1}$, and is a decreasing function of a_0 for all $W_{\max} > \frac{1}{q+1}$. Therefore, since $W_{\max} \leq \frac{q+3}{2q+4}$, $a_0 \leq M-h$, and $a_0 \geq 1$ (by Lemma 18), $f(a_0, W_{\max})$ is maximized when either $W_{\max} = \frac{q+3}{2q+4}$ and $a_0 = 1$ or $W_{\max} = \frac{1}{q+1}$ and $a_0 = M-h$. It can easily be verified that $f(a_0, \frac{q+3}{2q+4}) = a_0 \cdot \left(\frac{-q^2-2q+1}{2q+4}\right) + M \cdot \left(\frac{q+1}{2}\right) - h \cdot \left(\frac{q+1}{2}\right) < qM+1$ for all $a_0 \geq 1$. It can also be verified that $f(a_0, \frac{1}{q+1}) = \frac{M-h}{q+1} < qM+1$ for all a_0. Hence, $f(a_0, W_{\max}) < qM+1$, and therefore, $\mathsf{LAG}(\tau, t_h+1, \mathcal{S}) < qM+1$ holds.

Subcase 2(b): $\boldsymbol{W_{\max} \leq \frac{1}{2}}$. For this case, $(q+2)W_{\max} - 1 \leq q \cdot W_{\max}$ holds. Hence, by (52), we have

$$\begin{aligned}
&\mathsf{LAG}(\tau, t_h+1, \mathcal{S}) \\
&\quad < \quad a_0 \cdot W_{\max} + (a_{q-1} + a_q^1 + a_q^2) \cdot q \cdot W_{\max} \\
&\quad \leq \quad a_0 \cdot W_{\max} + (M-h-a_0) \cdot q \cdot W_{\max} \qquad \text{(by (32) because } a_q^0 = 0) \\
&\quad = \quad a_0 \cdot W_{\max}(1-q) + (M-h) \cdot q \cdot W_{\max} \\
&\quad \leq \quad (M-h) \cdot q \cdot W_{\max} \qquad \text{(because } q \geq 1) \\
&\quad < \quad qM+1.
\end{aligned}$$

By the reasoning in subcases 2(a) and 2(b), it follows that if $W_{\max} \leq \frac{q+3}{2q+4}$ and $a_0 \geq \frac{(M-h)\cdot(q+1)}{q+2}$, then $\mathsf{LAG}(\tau, t_h+1, \mathcal{S}) < qM+1$.

Finally, the lemma holds by the conclusions drawn in Cases 1 and 2. ■

By Lemmas 35 and 37, for any a_0 and $W_{\max}$, either $\mathsf{LAG}(\tau, t_h+1, \mathcal{S}) < qM+1$ or $\mathsf{LAG}(\tau', t_h-1, \mathcal{S}') \geq qM+1$ holds. Thus, either (T1) or Definition 5 is contradicted.

3.9. Case D: $(A_q^0 = A_q^1 = \emptyset)$

Lemma 38. *If $A_q^0 = A_q^1 = \emptyset$, then $\mathsf{LAG}(\tau, t_h+1) < qM+1$.*

Proof: Because $a_q^0 = a_q^1 = 0$, and a_0, a_{q-1}, and a_q^2 are independent of $W_{\max}$, as explained earlier (when (31) was established), we bound $\mathsf{LAG}(\tau, t_h+1)$ assuming $W_{\max} \geq 2/3$. Hence, by (31), and $A_q^0 = A_q^1 = \emptyset$, we have $\mathsf{LAG}(\tau, t_h+1) < a_0 \cdot W_{\max} + ((q+3)W_{\max} - 2) \cdot (a_q^2 + a_{q-1})$, which, by (32), equals $a_0 \cdot W_{\max} + ((q+3)W_{\max} - 2) \cdot (M-h-a_0)$.

Contrary to the statement of the lemma, assume $\mathsf{LAG}(\tau, t_h+1) \geq qM+1$. This assumption implies that $a_0 \cdot W_{\max} + ((q+3)W_{\max} - 2) \cdot (M-h-a_0) > qM+1$, which, in turn, implies that $W_{\max} > f \stackrel{\text{def}}{=} \frac{(q+2)M-2h-2a_0+1}{(q+3)M-(q+3)h-(q+2)a_0}$. We now determine a lower bound for f and show that f lies outside the range of values assumed for $W_{\max}$ and arrive at a contradiction. Let Y denote the denominator of f. The first derivative of f with respect to h is given by $\frac{q(q+3)M-2a_0+q+3}{Y^2}$, which is non-negative for all $M \geq 1$, $a_0 \geq 1$, and $q \geq 1$. The first derivative of f with respect to a_0 is given by $\frac{M(q^2+q-2)+2h+q+2}{Y^2}$, which is also non-negative for all $M \geq 1$, $q \geq 1$, and $h \geq 0$. Hence, since h and a_0 are greater than zero, f is minimized when $h = a_0 = 1$, for which $f = \frac{(q+2)M-3}{(q+3)M-2q-5} > \frac{q+2}{q+3}$ holds, for all $q \geq 1$, $M > 1$. This violates (W), and hence, our assumption is false, and the lemma follows. ■

By Lemmas 20, 27, 35, 37, and 38, if (W) is satisfied, then either $\mathsf{LAG}(\tau, t_h+1) < qM+1$ or there exists another task system with LAG under EPDF at least $qM+1$ at $t_h - 1$. Thus, either (T1) or the minimality of t_h is contradicted. So, task system τ as defined in Definition 6 does not exist, and Theorem 2 holds.

Theorem 2 implies that if each task weight is at most $W_{\max}$, then tardiness under EPDF is at most $\left\lceil \frac{3 \cdot W_{\max} - 2}{1 - W_{\max}} \right\rceil$, and we have the following corollary.

Corollary 1. *If the weight of each task in a feasible GIS task system τ is at most $W_{\max}$, then* EPDF *ensures a tardiness bound of* $\max(1, \left\lceil \frac{3 \cdot W_{\max} - 2}{1 - W_{\max}} \right\rceil)$ *for τ.*

Proof: Assume to the contrary that the tardiness for some subtask in τ is q, where $q > \max(1, \left\lceil \frac{3 \cdot W_{\max} - 2}{1 - W_{\max}} \right\rceil)$. Then, $q > \max(1, \frac{3 \cdot W_{\max} - 2}{1 - W_{\max}})$ holds, which implies that $q > 1$ and $W_{\max} < \frac{q+2}{q+3}$. This contradicts Theorem 2. ■

4. Conclusion

We have presented counterexamples that show that, in general, tardiness under the EPDF Pfair algorithm can exceed a small constant number of quanta for feasible recurrent real-time task systems. Thus, the conjecture that EPDF ensures a tardiness bound of one quantum for all feasible task systems is proved false. We have also presented sufficient per-task utilization restrictions that are more liberal than those previously known for ensuring a tardiness of q quanta under EPDF, where $q \geq 1$. EPDF is more efficient than known optimal Pfair algorithms and may be preferable for systems instantiated on less-powerful platforms, systems with soft timing constraints, and systems whose task composition can change at run-time.

For $q = 1$, our result presents an improvement of 50% over the previous one. This improvement is mainly due to the categorization of subtasks (presented in Section 3.1) and the ability to bound the number of miss initiators and successors of miss initiators scheduled in a slot with a hole (Lemmas 21–26), and the technique of relating the lag of a task system at a given time to that at an earlier time, developed for reasoning about Case C (presented in Section 3.8). Though we have not shown the per-task utilization restriction derived to be tight and do not believe it to be the case, we do believe that this result cannot be improved upon without adding significantly to the complexity of the analysis.

Acknowledgement. The first author is grateful to Anand Srinivasan for his EPDF simulator and for sharing the details of a task system that is close to missing a deadline by two quanta (specifically, whose LAG can be $M - 1$). Examination of that task system served as a starting point for the construction of several task systems that can miss their deadlines by two or more quanta, including those reported in this paper.

References

[1] J. Anderson and A. Srinivasan. Early-release fair scheduling. In *Proceedings of the 12th Euromicro Conference on Real-Time Systems*, pages 35–43, June 2000.

[2] J. Anderson and A. Srinivasan. Pfair scheduling: Beyond periodic task systems. In *Proceedings of the 7th International Conference on Real-Time Computing Systems and Applications*, pages 297–306, December 2000.

[3] J. Anderson and A. Srinivasan. Mixed Pfair/ERfair scheduling of asynchronous periodic tasks. *Journal of Computer and System Sciences*, 68(1):157–204, February 2004.

[4] S. Baruah, N. Cohen, C.G. Plaxton, and D. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, June 1996.

[5] U. Devi and J. Anderson. Improved conditions for bounded tardiness under EPDF fair multiprocessor scheduling. In *Proceedings of the 12th International Workshop on Parallel and Distributed Real-Time Systems*, April 2004. 8 pages (On CD-ROM).

[6] U. Devi and J. Anderson. A schedulable utilization bound for the multiprocessor EPDF Pfair algorithm. *Real-Time Systems*, 38(3):237–288, February 2008.

[7] K. Jeffay and S. Goddard. A theory of rate-based execution. In *Proceedings of the Real-Time Systems Symposium*, pages 304–314, Phoenix, AZ, December 1999. IEEE Computer Society Press.

[8] L. Sha, T. Abdelzaher, K.-E. Arzen, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A.K. Mok. Real time scheduling theory: A historical perspective. *Real-Time Systems*, 28(2/3):101–155, November/December 2004.
[9] A. Srinivasan. *Efficient and Flexible Fair Scheduling of Real-Time Tasks on Multiprocessors.* PhD thesis, University of North Carolina at Chapel Hill, December 2003.
[10] A. Srinivasan and J. Anderson. Optimal rate-based scheduling on multiprocessors. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 189–198, May 2002.
[11] A. Srinivasan and J. Anderson. Efficient scheduling of soft real-time applications on multiprocessors. In *Proceedings of the 15th Euromicro Conference on Real-Time Systems*, pages 51–59, July 2003.
[12] A. Srinivasan and J. Anderson. Fair scheduling of dynamic task systems on multiprocessors. *Journal of Systems and Software*, 77(1):67–80, April 2005.

A. Appendix: Proofs Omitted in the Main Text

In this appendix, we present all proofs omitted in the main paper. We begin with Claim 1.

A.1. Proof of Claim 1

Claim 1 *There is no hole in any slot in $[t_d - 1, t_d + q)$ in $\mathcal{S}'$.*

Proof: By Definition 3, (S1), and (S2), exactly one subtask in σ has a tardiness of $q + 1$. Let T_i denote that subtask. By (S1) again, the deadline of T_i is at t_d, and hence, T_i is scheduled at time $t_d + q$.

The proof of the claim is by induction on decreasing time t. We start by showing that there is no hole in slot $t_d + q - 1$.

Base Case: $t = t_d + q - 1$. Let T_h denote the predecessor, if any, of T_i. Because the deadlines of any two successive subtasks of the same task differ by at least one time unit, $\mathsf{d}(T_h) \le t_d - 1$ holds. Therefore, by Definition 3, the tardiness of T_h is at most q, and T_h completes executing by $t_d + q - 1$. Hence, no subtask of T is scheduled in slot $t_d + q - 1$. Thus, there is no hole in slot $t_d + q - 1$; otherwise, EPDF would schedule T_i there.

Induction Hypothesis. Assume that there is no hole in any slot in $[t', t_d + q)$, where $t_d - 1 < t' < t_d + q$.

Induction Step: $t = t' - 1$. We show that there is no hole in slot $t' - 1$. The deadline of every subtask scheduled in t' is at most t_d. Hence, the release time and the eligibility time of every such subtask is at or before $t_d - 1$. Since $t_d - 1 \le t' - 1$, every subtask scheduled at t' can be scheduled at $t' - 1$ unless its predecessor is scheduled there. By the induction hypothesis, there is no hole in slot t'. Hence, if there is a hole in $t' - 1$, then at most $M - 1$ of the M subtasks scheduled at t' can have their predecessors scheduled at $t' - 1$, implying that at least one of the subtasks scheduled at t' should have been scheduled at $t' - 1$, which is a contradiction. Therefore, there can be no hole in $t' - 1$. ■

A.2. Proofs from Section 3.1

Lemma 10 *The allocation received by a k-dependent subtask in its first slot in the ideal schedule are as follows.*

(a) *The allocation $\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{r}(T_i))$ received in the ideal schedule by a k-dependent subtask T_i of a* ***periodic*** *task T with $wt(T) < 1$ in the first slot of its window is at most $k \cdot \frac{T.e}{T.p} - (k-1) - \frac{1}{T.p}$, for all $k \ge 0$.*

(b) *The allocation $\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{r}(T_i))$ received in the ideal schedule by a k-dependent subtask T_i of a GIS task T in the first slot of its window is at most $k \cdot \frac{T.e}{T.p} - (k-1) - \frac{1}{T.p}$, for all $k \ge 0$.*

(c) *Let T_i, where $i \geq k{+}1$ and $k \geq 1$, be a subtask of T with $wt(T) < 1$ such that $|\omega(T_i)| \geq 3$ and $b(T_{i-1}) = 1$. Let the number of subtasks in T_{i-1}'s dependency group be at least k. Then, $\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{r}(T_i)) \leq k \cdot \frac{T.e}{T.p} - (k-1) - \frac{1}{T.p}$.*

Proof: Each part is proved below in turn.

Proof of part (a). The proof is by induction on k.

Base Case: $k = 0$. Because $wt(T) < 1$, and $T.e$ and $T.p$ are integral, $T.e \leq T.p - 1$. Thus, by (9), $\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{r}(T_i)) \leq wt(T) = T.e/T.p \leq (T.p-1)/T.p = 1 - 1/T.p$, and the lemma holds for the base case.

Induction Step. Assuming that the lemma holds for $(k-1)$-dependent subtasks, we show that it holds for k-dependent subtasks, where $k \geq 1$. Because $k \geq 1$, by the definition of k-dependency, $i > 1$ and T is heavy. Hence, by Lemma 1, $|\omega(T_{i-1})|$ is either two or three. We consider two cases.

Case 1: $|\omega(T_{i-1})| = 2$. Since $k \geq 1$, T_{i-1} is $(k-1)$-dependent. Therefore, by the induction hypothesis,

$$\mathsf{A}(\mathsf{ideal}, T_{i-1}, \mathsf{r}(T_{i-1})) \leq (k-1) \cdot (T.e/T.p) - (k-2) - (1/T.p). \tag{54}$$

Because $|\omega(T_{i-1})| = 2$, by (8), $\mathsf{A}(\mathsf{ideal}, T_{i-1}, \mathsf{d}(T_{i-1}) - 1) = 1 - \mathsf{A}(\mathsf{ideal}, T_{i-1}, \mathsf{r}(T_{i-1}))$. Hence, by (54), $\mathsf{A}(\mathsf{ideal}, T_{i-1}, \mathsf{d}(T_{i-1}) - 1) \geq (k-1) + (1/T.p) - (k-1) \cdot (T.e/T.p)$. Because T_i is k-dependent, where $k \geq 1$, by Lemma 8(c), $b(T_{i-1}) = 1$, and by Lemma 3, $\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{r}(T_i)) = (T.e/T.p) - \mathsf{A}(\mathsf{ideal}, T_{i-1}, \mathsf{d}(T_{i-1}) - 1) \leq k \cdot (T.e/T.p) - (k-1) - (1/T.p)$.

Case 2: $|\omega(T_{i-1})| = 3$. By the contra-positive of Lemma 8(c), T_{i-1} is 0-dependent; hence, T_i is 1-dependent, *i.e.*, $k = 1$. By Lemma 8(c), $\mathsf{b}(T_{i-1}) = 1$, and hence, by Lemma 3,

$$\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{r}(T_i)) \;=\; \frac{T.e}{T.p} - \mathsf{A}(\mathsf{ideal}, T_i, \mathsf{d}(T_{i-1}) - 1) \;\leq\; \frac{T.e}{T.p} - \frac{1}{T.p} \qquad \text{(by 10).}$$

■

Proof of part (b). Follows from part (a) and the definition of GIS tasks. (The allocation that T_i receives in each slot of its window is identical to the allocation that it would receive if T were periodic.) ■

Proof of part (c). Since $|\omega(T_i)| \geq 3$, by Lemma 8(c), T_i is 0-dependent and is the first subtask in its group. Hence, T_{i-1} is the final subtask in its dependency group, and since there are at least k subtasks in T_{i-1}'s group, T_{i-1} is at least $(k-1)$-dependent. Hence, by Lemma 10(b), $\mathsf{A}(\mathsf{ideal}, T_{i-1}, \mathsf{r}(T_{i-1})) \leq (k-1) \cdot \frac{T.e}{T.p} - (k-2) - \frac{1}{T.p}$. (What follows is similar to the reasoning used in the induction step in the proof of Lemma 10(a).) If $|\omega(T_{i-1})| = 2$, then, by (8), $\mathsf{A}(\mathsf{ideal}, T_{i-1}, \mathsf{d}(T_{i-1}) - 1) \geq 1 - ((k-1) \cdot \frac{T.e}{T.p} - (k-2) - \frac{1}{T.p}) = (k-1) - (k-1) \cdot \frac{T.e}{T.p} + \frac{1}{T.p}$. By the statement of the lemma, $b(T_{i-1}) = 1$, and hence, by Lemma 3, $\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{r}(T_i)) = wt(T) - \mathsf{A}(\mathsf{ideal}, T_{i-1}, \mathsf{d}(T_{i-1}) - 1) \leq k \cdot \frac{T.e}{T.p} - (k-1) - \frac{1}{T.p}$. Thus, the lemma holds when $|\omega(T_{i-1})| = 2$.

On the other hand, if $|\omega(T_{i-1})| \geq 3$, then by Lemma 8(c), T_{i-1} is 0-dependent. By (10), $\mathsf{A}(\mathsf{ideal}, T_{i-1}, \mathsf{d}(T_{i-1}) - 1) \geq \frac{1}{T.p}$, and hence, because $b(T_{i-1}) = 1$, by Lemma 3, $\mathsf{A}(\mathsf{ideal}, T_i, \mathsf{r}(T_i)) = wt(T) - \mathsf{A}(\mathsf{ideal}, T_{i-1}, \mathsf{d}(T_{i-1})) \leq \frac{T.e}{T.p} - \frac{1}{T.p}$. By the statement of the lemma, $|\omega(T_i)| = 3$, and hence, T_i is also 0-dependent. Thus, T_{i-1} is the only subtask in its group, and hence, $k = 1$. (Note that k here denotes the number of subtasks that are in the same dependency group as T_{i-1}.) Therefore, the lemma holds for this case too. ■

Lemma 11 *Let T_i be a k-dependent subtask of a task T for $k \geq 0$, and let the tardiness of T_i be s for some $s \geq 1$ (that is, T_i is scheduled at time $\mathsf{d}(T_i) + s - 1$). Then $\mathsf{lag}(T, \mathsf{d}(T_i) + s) < (k + s + 1) \cdot wt(T) - k$.*

Proof: By the statement of the lemma, T_i and all prior subtasks of T are scheduled in $[0, \mathsf{d}(T_i) + s)$. Hence, $\mathsf{lag}(T, \mathsf{d}(T_i) + s)$ depends on the number of subtasks of T after T_i released prior to $\mathsf{d}(T_i) + s$, the allocations they receive in the ideal schedule, and when they are scheduled in $\mathcal{S}$. It can be verified from (1) and (2) that at most $s+1$ successors of T_i — $T_{i+1}, \ldots, T_{i+s+1}$ — are released before $\mathsf{d}(T_i)+s$. Hence, the lag of T at $\mathsf{d}(T_i)+s$ in $\mathcal{S}$ is maximized if all those subtasks are present and are released without any IS separations and $\mathcal{S}$ has not scheduled any of them by time $\mathsf{d}(T_i)+s$. We will assume that this is the case. (The statement of the lemma implies that none of those subtasks is scheduled by $\mathsf{d}(T_i)+s$.) By Lemma 2, at most one successor of T_i, namely T_{i+1}, can have a release time that is before $\mathsf{d}(T_i)$. Further, $\mathsf{r}(T_{i+1}) \geq \mathsf{d}(T_i)-1$ holds. Hence, $\mathsf{lag}(T, \mathsf{d}(T_i)+s) \leq \mathsf{A}(\mathsf{ideal}, T_{i+1}, \mathsf{d}(T_i)-1)) + \mathsf{A}(\mathsf{ideal}, T, \mathsf{d}(T_i), \mathsf{d}(T_i)+s)$. If $\mathsf{r}(T_{i+1}) > \mathsf{d}(T_i)-1$ holds, then $\mathsf{A}(\mathsf{ideal}, T_{i+1}, \mathsf{d}(T_i)-1) = 0$. On the other hand, if $\mathsf{r}(T_{i+1}) = \mathsf{d}(T_i)-1$, then by (4), $\mathsf{b}(T_i) = 1$. Further, either $|\omega(T_{i+1})| = 2$ or $|\omega(T_{i+1})| > 2$. In the former case, T is heavy, and because $b(T_i) = 1$, by the definition of k-dependency (and given by Lemma 8(a)), T_{i+1} belongs to the same dependency group as T_i and is $(k+1)$-dependent. Hence, by Lemma 10(b), $\mathsf{A}(\mathsf{ideal}, T_{i+1}, \mathsf{r}(T_{i+1})) \leq (k+1) \cdot wt(T) - k - \frac{1}{T.p}$. If the latter holds, *i.e.*, $|\omega(T_{i+1})| > 2$, we reason as follows. Since T_i is k-dependent, the number of subtasks in T_i's group is at least $k+1$. Therefore, since $\mathsf{b}(T_i) = 1$, Lemma 10(c) applies for T_{i+1} and it follows that $\mathsf{A}(\mathsf{ideal}, T_{i+1}, \mathsf{r}(T_{i+1})) \leq (k+1) \cdot wt(T) - k - \frac{1}{T.p}$. Thus, in either case, $\mathsf{A}(\mathsf{ideal}, T_{i+1}, \mathsf{d}(T_i)-1) = \mathsf{A}(\mathsf{ideal}, T_{i+1}, \mathsf{r}(T_{i+1})) \leq (k+1) \cdot wt(T) - k - \frac{1}{T.p}$.

By (7), $\mathsf{A}(\mathsf{ideal}, T, \mathsf{d}(T_i), \mathsf{d}(T_i)+s) \leq s \cdot wt(T)$. Hence, $\mathsf{lag}(T, \mathsf{d}(T_i)+s) \leq \mathsf{A}(\mathsf{ideal}, T_{i+1}, \mathsf{d}(T_i)-1)) + \mathsf{A}(\mathsf{ideal}, T, \mathsf{d}(T_i), \mathsf{d}(T_i)+s) \leq (k+s+1) \cdot wt(T) - k - \frac{1}{T.p} < (k+s+1) \cdot wt(T) - k$. ■

A.3. Proofs from Section 3.4

Lemma 18 *There exists a subtask W_ℓ scheduled at t_h with $\mathsf{e}(W_\ell) \leq t_b$, $\mathsf{d}(W_\ell) = t_h + 1$, and $\mathcal{S}(W, t) = 0$, for all $t \in [t_b, t_h)$. Also, there is no hole in any slot in $[t_b, t_h)$. (Note that, by this lemma, $A_0(t_h) \neq \emptyset$.)*

Proof: We first show that the first subtask to be displaced upon U_j's removal (where U_j is as defined in Def. 8) has properties as stated for W_ℓ, *i.e.*, is eligible at or before t_b and has its deadline at $t_h + 1$.

Let τ' be the task system obtained by removing U_j from τ, and let $\mathcal{S}'$ be the EPDF schedule for τ'. Let $\Delta_1 = \langle X^{(1)}, t_1, X^{(2)}, t_2 \rangle$ be the first valid displacement, if any, that results due to the removal of U_j. Then, $X^{(1)} = U_j$, $t_1 = t_b$, and by Lemma 5,

$$t_2 > t_1 = t_b. \quad (55)$$

We first show that $t_2 \geq t_h$.

Assume to the contrary that $t_2 < t_h$. Then, by (55) and Definition 7, T is not in $B(t_h)$. Therefore, T is in $I(t_h)$ or in $A(t_h)$. In either case,

$$\mathsf{d}(X^{(2)}) \leq t_h. \quad (56)$$

To see this, note that if $T \in I(t_h)$, then because T is not active at t_h, by Definition 1, $\mathsf{d}(X^{(2)}) \leq t_h$. On the other hand, if $T \in A(t_h)$, then consider T's subtask, say T_k, scheduled at t_h. By Lemma 16, $\mathsf{d}(T_k) \leq t_h + 1$. Because $X^{(2)}$ is scheduled at $t_2 < t_h$, $X^{(2)}$ is an earlier subtask of T than T_k, and hence, by (1) and (2), $\mathsf{d}(X^{(2)}) \leq t_h$. Because is U_j is U's critical subtask at t_h and U is in $B(t_h)$, by Lemma 17, we have

$$\mathsf{d}(U_j) = t_h + 1. \quad (57)$$

By (56) and (57), $\mathsf{d}(U_j) > \mathsf{d}(X^{(2)})$. However, since EPDF selects U_j over $X^{(2)}$ at time t_b (which follows because the displacement under consideration is valid), this is a contradiction. Thus, our assumption that $t_2 < t_h$ holds is false.

Having shown that $t_2 \geq t_h$, we next show $t_2 = t_h$. Assume, to the contrary, that $t_2 > t_h$. Since displacement $\Delta_1 = \langle U_j, t_b, T_i, t_2 \rangle$is valid, $\mathsf{e}(X^{(2)}) \leq t_b$. This implies that $X^{(2)}$ is eligible to be scheduled at t_h (*i.e.*, T is not scheduled at t_h), and because there is a hole in t_h, it should have been scheduled there in $\mathcal{S}$, and not later at t_2. It follows that $t_2 = t_h$.

Finally, because U_j is scheduled at t_b in preference to $X^{(2)}$, $\mathsf{d}(T_i) \geq \mathsf{d}(U_j) = t_h + 1$ (from (57)), which by Lemma 15 (since $X^{(2)}$ is scheduled in slot t_h) implies that

$$\mathsf{d}(X^{(2)}) = t_h + 1. \tag{58}$$

Thus, the first subtask, if any, to be displaced upon U_j's removal satisfies the properties specified for W_ℓ in the statement of the lemma. Hence, if a subtask with such properties does not exist, then U_j's removal will not lead to any displacements.

Next, we show that unless the other two conditions specified in the lemma also hold, no subtask will be displaced upon U_j's removal. For this, first note that by (57) and (58) $X^{(2)}$ and U_j have equal deadlines, and hence, $X^{(2)}$ is not U_j's successor. Next, note that because $\langle U_j, t_b, X^{(2)}, t_h \rangle$ is valid, no subtask of T prior to $X^{(2)}$ is scheduled in $[t_b, t_h)$, and also if there is a hole in any slot t in $[t_b, t_h)$, then EPDF would have scheduled $X^{(2)}$ at t.

Thus, if the lemma is false, then removing U_j does not result in any displacements. We now show that, in such a case, $\mathsf{LAG}(\tau', t_h + 1, \mathcal{S}') \geq qM + 1$. $\mathsf{LAG}(\tau', t_h + 1, \mathcal{S}') = \mathsf{A}(\mathsf{ideal}, \tau', 0, t_h + 1) - \mathsf{A}(\mathcal{S}', \tau', 0, t_h + 1)$. τ' contains every subtask that is in τ except U_j. U_j is scheduled before t_h in $\mathcal{S}$, and by (57), $\mathsf{d}(U_j) = t_h + 1$. Therefore, U_j receives an allocation of one quantum by time $t_h + 1$ in the ideal schedule for τ, and hence, $\mathsf{A}(\mathsf{ideal}, \tau', 0, t_h + 1) = \mathsf{A}(\mathsf{ideal}, \tau, 0, t_h + 1) - 1$. Similarly, since no subtask other than U_j of τ is displaced or removed in $\mathcal{S}'$, the total allocation to τ' in $\mathcal{S}'$ up to time $t_h + 1$, $\mathsf{A}(\mathcal{S}', \tau', 0, t_h + 1)$, is $\mathsf{A}(\mathcal{S}, \tau, 0, t_h + 1) - 1$. Therefore, $\mathsf{LAG}(\tau', t_h + 1, \mathcal{S}') = \mathsf{A}(\mathsf{ideal}, \tau, 0, t_h + 1) - \mathsf{A}(\mathcal{S}, \tau, 0, t_h + 1) = \mathsf{LAG}(\tau, t_h + 1, \mathcal{S}) \geq qM + 1$ (by (T1)). To conclude, we have shown that, τ' with one fewer subtask than τ also has a LAG of at least $qM + 1$ at $t_h + 1$, which contradicts (T2). ■

Lemma 19 *Let $t_m \leq t_h$ be a slot in which an MI is scheduled. Then, the following hold.*

(**a**) *For all t, where $t_m - (q+2) < t < t_m$, there is no hole in slot t, and for each subtask V_k that is scheduled in t, $\mathsf{d}(V_k) \leq t_m - q + 1$.*

(**b**) *Let W be a task in $B(t_m)$ and let the critical subtask W_ℓ of W at t_m be scheduled before t_m. Then, W_ℓ is scheduled at or before $t_m - (q + 2)$.*

Proof of part (a). The proof is by induction on decreasing t. We start with $t = t_m - 1$.

Base Case: $\boldsymbol{t = t_m - 1}$. Let T_i be an MI scheduled at t_m. (By the statement of the lemma, at least one MI is scheduled in t_m.) Then, $\mathsf{d}(T_i) = t_m - q + 1$, and $\mathcal{S}(T, t_m - 1) = 0$, from the definition of an MI. Hence, T_i is eligible at $t_m - 1$. Because T_i is not scheduled at $t_m - 1$, it follows that there is no hole in $t_m - 1$ and that the priority of every subtask V_k scheduled at $t_m - 1$ is at least that of T_i, *i.e.*, $\mathsf{d}(V_k) \leq \mathsf{d}(T_i) = t_m - q + 1$.

Induction Hypothesis. Assume that the claim in part (a) holds for all t, where $t' + 1 \leq t \leq t_m - 1$ and $t_m - (q + 1) < t' + 1 < t_m$.

Induction Step. We now show that the claim holds for $t = t'$. By the induction hypothesis, there is no hole in $t' + 1$ and $\mathsf{d}(T_i) \leq t_m - q + 1$ holds for every subtask T_i scheduled in $t' + 1$. Therefore, since $wt(T) < 1$, by (1), $\mathsf{r}(T_i) \leq t_m - q - 1$. Thus, there are M subtasks with a release time at or before $t_m - q - 1$ and deadline at or before $t_m - q + 1$ scheduled at $t' + 1 \geq t_m - q$. If there is either a hole in t' or a subtask with deadline later than $t_m - q + 1$ scheduled in t', then there is at least one subtask scheduled in $t' + 1$ whose predecessor is not scheduled in t'. Such a subtask is eligible at t', since its release time is at or before $t_m - q - 1 \leq t'$. Hence, if there is a hole in t', then the work-conserving behavior of EPDF is contradicted. Otherwise, the pseudo-deadline-based scheduling of EPDF is contradicted. Hence, the claim holds for $t = t'$. ■

Proof of part (b). By Definition 2, $\mathsf{d}(W_\ell) \geq t_m + 1$. Hence, since $q \geq 1$, this part easily follows from part (a). ■

A.4. Proofs from Section 3.8

Lemma 28 *The following properties hold for subsets τ_s^i and τ^i defined in Section 3.8, where $1 \le i \le 8$.*

(**a**) *For every task T, there is at most one subtask in $(\tau_s^1 \cup \tau_s^2 \cup \tau_s^6)$.*

(**b**) *Let T_i scheduled at t_h be the subtask of a task T in $A_q(t_h)$ or $A_{q-1}(t_h)$. Then, T_i is in τ_s^5.*

(**c**) *$\tau^7 \subseteq \tau^1$ and $\tau^8 \subseteq \tau^2$.*

(**d**) *Subsets τ^i, where $1 \le i \le 6$, are pairwise disjoint.*

Proof: Each of the above properties is proved below.

Proof of part (a). We first show that each task T has at most one subtask in τ_s^1. Let T_i in τ_s^1 be the critical subtask at t_h of T, which is in $B(t_h)$. Then, by Lemma 17, $\mathsf{d}(T_i) = t_h + 1$ holds. Because $wt(T) < 1$, by (1), $\mathsf{r}(T_i) \le \mathsf{d}(T_i) - 2 = t_h - 1$ holds. Hence, by the definition of a critical subtask in Definition 2, T_i is critical at $t_h - 1$ also. Thus, if T has a critical subtask T_i at t_h and T is in $B(t_h)$, then T cannot have a subtask that is different from T_i that is critical at $t_h - 1$. Hence, it follows that each task has at most one subtask in τ_s^1.

We next show that each task can have at most one subtask in $\tau_s^2 \cup \tau_s^6$. Note that a subtask is in τ_s^2 or τ_s^6 only if it is scheduled at $t_h - 1$. Further, each task T can have at most one subtask scheduled at $t_h - 1$. Hence, if T's subtask T_i scheduled at $t_h - 1$ has its deadline at or after t_h, then T_i is in τ_s^2; else, in τ_s^6.

Finally, we show that if T has a subtask T_i in τ_s^1, then it does not have a subtask in $\tau_s^2 \cup \tau_s^6$, and vice versa. If T_i is in $B(t_h - 1)$, then T cannot have a subtask scheduled at $t_h - 1$, and hence, cannot have a subtask in $\tau_s^2 \cup \tau_s^6$ (because every subtask in these sets is scheduled at $t_h - 1$). On the other hand, if T_i is in $B(t_h)$ and is T's critical subtask at t_h, then note the following. **(i)** τ_s^1 is non-empty only if t_b' exists; **(ii)** by Lemma 17, $\mathsf{d}(T_i) = t_h + 1$ holds; and **(iii)** T_i is scheduled at or before t_b', whereas a subtask in $\tau_s^2 \cup \tau_s^6$ is scheduled at $t_h - 1$. By (44), $t_b' \le t_h - (q + 3)$. Thus, by (ii) and (iii), no subtask of T with a deadline at or before t_h can be scheduled at $t_h - 1$, and hence, can be in $\tau_s^2 \cup \tau_s^6$. On the other hand, if a subtask of T with a deadline after t_h is scheduled at $t_h - 1$, then it contradicts the fact that T_i is T's critical subtask at t_h. So, no such subtask can be in $\tau_s^2 \cup \tau_s^6$ either. ■

Proof of part (b). By the conditions of Case C, no c-MI, where $c > 0$, is scheduled at t_h. Further, because T is in $A_q(t_h)$ or $A_{q-1}(t_h)$, tardiness of T_i is greater than zero. Hence, by the definition of c-MI and because T is not a c-MI, T is also scheduled at $t_h - 1$. Therefore, T_i is in τ_s^5. ■

Proof of part (c). Immediate from the definitions. ■

Proof of part (d). By part (a), every task T has at most one subtask in $\tau_s^1 \cup \tau_s^2 \cup \tau_s^6$. Therefore, τ^1, τ^2, and τ^6 are pairwise disjoint. By (25) and (26), A_0, A_q, and A_{q-1} are pairwise disjoint, and hence, by their definitions, τ_s^3, τ_s^4, and τ_s^5 are pairwise disjoint, and subtasks in them are scheduled at t_h. However, by the definitions of τ_s^1, τ_s^2, and τ_s^6, no task of a subtask in any of these subsets is scheduled at t_h. Therefore, a task in τ^1, τ^2, or τ^6 is not in $\cup_{i=3}^5 \tau^i$, that is $\tau^1 \cup \tau^2 \cup \tau^6$ is disjoint from $\cup_{i=3}^5 \tau^i$. Since τ^1, τ^2, and τ^6 are pairwise disjoint, as are τ^3, τ^4, and τ^5, all six subsets are pairwise disjoint. ■

We make the following two claims before proving the remaining lemmas.

Claim 4. *No subtask with deadline at or before $t_h - 1$ is removed or displaced in $\mathcal{S}'$.*

Proof: Follows from the fact that the deadline of every subtask removed, that is, the deadline of every subtask in τ_s^R (refer 45), is at or after t_h. Hence, because ties in $\mathcal{S}$ and $\mathcal{S}'$ are resolved identically, the removed subtasks cannot impact how subtasks with earlier deadlines are scheduled, and hence, cannot cause such subtasks to be displaced. (Subtasks in τ_s^1 are critical subtasks at t_h or at $t_h - 1$, and hence their deadlines are at or after t_h. Similarly, subtasks in τ_s^3 are scheduled at t_h and have a tardiness of zero, implying that their deadlines are at or after $t_h + 1$.) ■

Claim 5. *The release time of every subtask in τ is at or before t_h.*

Proof: Because there is a hole in t_h (by (H)), by Lemma 15, no subtask scheduled at or before t_h can have a deadline after $t_h + 1$, implying that the release time of every such subtask is at or before t_h. Hence, a subtask with release time after t_h is scheduled after t_h in $\mathcal{S}$. For every such subtask, allocations in both the ideal schedule and $\mathcal{S}$ are zero in $[0, t_h + 1)$. Therefore, the LAG of τ at $t_h + 1$ does not depend on such a subtask. Further, if such a subtask is removed, the schedule before $t_h + 1$ is not impacted and no subtask scheduled at or after $t_h + 1$ can shift to t_h or earlier. Hence, the LAG of τ at $t_h + 1$ is not altered. Thus, the presence of subtasks released after t_h contradicts (T2). ■

Lemma 29 *Let T be a task with a subtask in τ_s^1 or τ_s^2. Then, $\mathsf{lag}(T, t_h - 1, \mathcal{S}') = \mathsf{lag}(T, t_h + 1, \mathcal{S})$.*

Proof: By (11),

$$\mathsf{lag}(T, t_h - 1, \mathcal{S}') = \mathsf{A}(\mathsf{ideal}_{\tau'}, T, 0, t_h - 1) - \mathsf{A}(\mathcal{S}', T, 0, t_h - 1). \tag{59}$$

To prove this lemma, we will express the allocation to T in $\mathsf{ideal}_{\tau'}$ and $\mathcal{S}'$ in terms of its allocations in ideal_{τ} and $\mathcal{S}$, respectively. We will establish some properties needed for this purpose.

By Lemma 28(a), T has exactly one subtask in $\tau_s^1 \cup \tau_s^2$. Let T_i denote the distinct subtask of T that is in τ_s^1 or τ_s^2, and T_j, its predecessor in τ_s^7 or τ_s^8, respectively, if any. Note that T_j does not exist if $\mathsf{d}(T_i) = t_h$, and need not necessarily exist otherwise.

Regardless of whether T_i is in τ_s^1 or τ_s^2, T_i is scheduled at or before t'_b in $\mathcal{S}$, which by (44), is before $t_h - 1$. Hence, because there is a hole in t_h, by Lemma 15, $\mathsf{d}(T_i) \le t_h + 1$ holds. We next show that the following holds.

(D) No subtask of T has its deadline after $t_h + 1$.

Since T is not scheduled in t_h and there is a hole in t_h, T_i's successor, if any, cannot have its eligibility time at or before t_h and deadline after $t_h + 1$. By Claim 5, no subtask in τ has a release time at or after $t_h + 1$. Thus, (D) holds.

We next claim that of T's subtasks, only T_i and/or T_j may receive non-zero allocations in the ideal schedule for τ in slots $t_h - 1$ and/or t_h. For this, note that the following hold: **(i)** since $\mathsf{d}(T_j) = t_h$ (by the definitions of τ_s^7 and τ_s^8), no subtask of T prior to T_j has its deadline after $t_h - 1$; **(ii)** because there is a hole in t_h, and T is not scheduled at t_h in $\mathcal{S}$ (by the definitions of τ_s^1 and τ_s^2), no subtask of T released after T_i has its eligibility time, and hence, release time at or before t_h. Hence, by (6), no subtask of T other than T_i and T_j receives any allocation in $t_h - 1$ and/or t_h. By (i) and (ii) above and because τ' contains every subtask of T that is in τ except T_i and T_j, we have $\mathsf{A}(\mathsf{ideal}_{\tau'}, T, 0, t_h - 1) = \mathsf{A}(\mathsf{ideal}_{\tau}, T, 0, t_h + 1) - \mathsf{A}(\mathsf{ideal}_{\tau}, T_i, 0, t_h + 1) - \mathsf{A}(\mathsf{ideal}_{\tau}, T_j, 0, t_h + 1)$. Because the deadlines of T_i and T_j are at most $t_h + 1$, both these subtasks receive ideal allocations of one quantum each by $t_h + 1$. Hence,

$$\mathsf{A}(\mathsf{ideal}_{\tau'}, T, 0, t_h - 1) = \begin{cases} \mathsf{A}(\mathsf{ideal}_{\tau}, T, 0, t_h + 1) - 2, & \text{if } T_j \text{ exists} \\ \mathsf{A}(\mathsf{ideal}_{\tau}, T, 0, t_h + 1) - 1, & \text{if } T_j \text{ does not exist.} \end{cases} \tag{60}$$

We now express the allocation to T in $\mathcal{S}'$ in terms of its allocation in $\mathcal{S}$. If T_i is in τ_s^1, then, in $\mathcal{S}$, T_i is scheduled at or before $t'_b \le t_h - (q + 3) \le t_h - 1$ (refer (44)); if it is in τ_s^2, then T_i is scheduled at $t_h - 1$.

Thus, in either, case T_i is scheduled at or before $t_h - 1$ in $\mathcal{S}$. Hence, T_j, if it exists, is scheduled at or before $t_h - 1$ in $\mathcal{S}$. As for where other subtasks of T are scheduled in $\mathcal{S}$, there is a hole in t_h, and (by the definitions of τ_s^1 and τ_s^2) T is not scheduled at t_h. Therefore, if some subtask of T is scheduled after t_h, then its eligibility time is at or after $t_h + 1$, and hence its deadline is after $t_h + 1$. However, by (D), no subtask of T has a deadline after $t_h + 1$. Hence, there does not exist a subtask of T that is scheduled after t_h in $\mathcal{S}$, which implies that there does not exist a subtask of T that is scheduled after t_h in $\mathcal{S}$ and before $t_h - 1$ in $\mathcal{S}'$. Further, because no subtask can displace to the right, there does not exist a subtask of T that is scheduled before $t_h - 1$ in $\mathcal{S}$, and at or after $t_h - 1$ in $\mathcal{S}'$. As already mentioned, every subtask of T except T_i and T_j is present in τ'. Therefore,

$$\mathsf{A}(\mathcal{S}', T, 0, t_h - 1) = \begin{cases} \mathsf{A}(\mathcal{S}, T, 0, t_h + 1) - 2, & \text{if } T_j \text{ exists} \\ \mathsf{A}(\mathcal{S}, T, 0, t_h + 1) - 1, & \text{if } T_j \text{ does not exist.} \end{cases} \tag{61}$$

By (59)–(61), regardless of whether T_j exists, $\mathsf{lag}(T, t_h - 1, \mathcal{S}') = \mathsf{A}(\mathsf{ideal}_\tau, T, 0, t_h + 1) - \mathsf{A}(\mathcal{S}, T, 0, t_h + 1) = \mathsf{lag}(T, t_h + 1, \mathcal{S})$. ■

Lemma 30 *Let T be a task with a subtask in τ_s^3. Then,* $\mathsf{lag}(T, t_h - 1, \mathcal{S}') > \mathsf{lag}(T, t_h + 1, \mathcal{S}) - 1/(q + 2)$.

Proof: Let T_i be T's subtask in τ_s^3. In $\mathcal{S}$, T_i is scheduled at t_h and is ready at or before $t_h - (q + 3)$. Therefore, by Lemma 7(a), $\mathsf{r}(T_i) \leq t_h - (q + 3)$ holds. Since T is in $A_0(t_h)$, and T_i is scheduled at t_h in $\mathcal{S}$, the tardiness of T_i is zero in $\mathcal{S}$. Therefore, $\mathsf{d}(T_i) \geq t_h + 1$ holds, which by (H) and Lemma 16 implies that

$$\mathsf{d}(T_i) = t_h + 1. \tag{62}$$

Hence, $|\omega(T_i)| = \mathsf{d}(T_i) - \mathsf{r}(T_i) \geq q + 4$ holds, and using Lemma 1, it can be shown that $wt(T) < 1/(q + 2)$. By Lemma 12(c), $\mathsf{lag}(T, t_h + 1, \mathcal{S}) < wt(T)$, and hence, because $wt(T) < 1/(q + 2)$, it follows that

$$\mathsf{lag}(T, t_h + 1, \mathcal{S}) < 1/(q + 2). \tag{63}$$

We next show that $\mathsf{lag}(T, t_h - 1, \mathcal{S}') = 0$. For this, we need to show that the total allocation to T in $[0, t_h - 1)$ is equal in $\mathsf{ideal}_{\tau'}$ and $\mathcal{S}'$. We first show that the total allocation in $[0, t_h - 1)$ to subtasks of T released after T_i is zero in both $\mathcal{S}'$ and $\mathsf{ideal}_{\tau'}$. By (62) and Lemma 2, the release time of the successor, T_j, if any, of T_i is at or after t_h. Hence, the allocation to every subtask of T released after T_i is zero in $[0, t_h - 1)$ in the ideal schedule for τ'. Also, because T_i is scheduled at t_h in $\mathcal{S}$, T_j is scheduled at or after $t_h + 1$ in $\mathcal{S}$. Hence, by Lemma 7(a), $\mathsf{e}(T_j) \geq t_h$ holds. Therefore, every subtask of T released after T_i is scheduled at or after t_h in $\mathcal{S}'$, that is, receives zero allocation in $[0, t_h - 1)$ in $\mathcal{S}'$.

We now show that subtasks of T released before T_i receive equal allocations in $[0, t_h - 1)$ in both $\mathsf{ideal}_{\tau'}$ and $\mathcal{S}'$. Since T_i is ready at or before $t_h - (q + 3)$, T_i's predecessor, if any, and all prior subtasks of T, if any, complete executing at or before $t_h - (q + 3)$ in $\mathcal{S}$, and hence, in $\mathcal{S}'$, as well (because no subtask can displace to the right). Furthermore, as discussed above, $\mathsf{r}(T_i) \leq t_h - (q + 3)$ holds, and hence, by Lemma 2, the deadline of T_i's predecessor is at or before $t_h - (q + 2)$. Hence, all subtasks released before T_i complete executing by $t_h - (q + 2)$ in $\mathsf{ideal}_{\tau'}$ as well.

Therefore, because T_i is not present in τ', the total allocation to all the subtasks of T in τ' in $[0, t_h - 1)$ is equal in $\mathcal{S}'$ and $\mathsf{ideal}_{\tau'}$. Hence, $\mathsf{lag}(T, t_h - 1, \mathcal{S}') = 0$, and because (63) holds, the lemma follows. ■

Lemma 31 *Let T be a task with a subtask in τ_s^4. Then,* $\mathsf{lag}(T, t_h - 1, \mathcal{S}') \geq \mathsf{lag}(T, t_h + 1, \mathcal{S}) - 2 \cdot W_{\max} + 1$.

Proof: First, we show that (R) below holds.

(R) No subtask of T is removed.

For this, note that because T is in τ^4, by Lemma 28(d), T is not in τ^i, where $1 \leq i \leq 6$ and $i \neq 4$. Hence, by Lemma 28(c), T is also not in τ^7 or in τ^8. Thus, T does not have a subtask in τ_s^R, and hence, (R) holds.

Let T_i be T's subtask in τ_s^4 and let $t_c = t_h - (q+3)$. Then, T_i is not ready at t_c in $\mathcal{S}$. We show that T_i is not ready at t_c in $\mathcal{S}'$ also. Let T_j denote T_i's predecessor, if any, in τ.

We now show that no subtask of T that is scheduled at or after $t_h - 1$ in $\mathcal{S}$ is scheduled before $t_h - 1$ in $\mathcal{S}'$. Note that T_i is scheduled at t_h in $\mathcal{S}$. Hence, it suffices to show that T_i is not scheduled before $t_h - 1$ in $\mathcal{S}'$ (which would imply that no later subtask is scheduled before $t_h - 1$), and if T_j is scheduled at $t_h - 1$ in $\mathcal{S}$, then it is not scheduled earlier in $\mathcal{S}'$.

Because T_i is scheduled at t_h in $\mathcal{S}$ and T_i is not ready at t_c in $\mathcal{S}$, Lemma 7(a) implies that either $\mathsf{r}(T_i) > t_c$, or $\mathsf{r}(T_i) \leq t_c$ and T_j exists and does not complete executing by t_c. If the former holds, then because $\mathsf{r}(T_i) > t_c$ and T_i is scheduled at $t_h > t_h - (q+3) = t_c$ in $\mathcal{S}$, by Lemma 7(a), $\mathsf{e}(T_i) > t_c$ holds, and hence, T_i is not eligible, and hence, not ready, at t_c in $\mathcal{S}'$ either. If the latter holds, then by Lemma 2, $\mathsf{d}(T_j) \leq t_c + 1 \leq t_h - (q+2)$ holds, and hence, by Claim 4, T_j is not displaced, and does not complete executing by t_c in $\mathcal{S}'$ also. Therefore, T_i is not ready at t_c in this case too.

Given that T_i is not ready at t_c in $\mathcal{S}'$, it is easy to show that T_i is not scheduled before $t_h - 1$ in $\mathcal{S}'$. For this, note that by Claim 4, no subtask with deadline at or before $t_h - 1$ is displaced or removed. Hence, since (C) holds, no subtask scheduled in $[t_h - (q+2), t_h - 1)$ is displaced or removed. Therfore, because T_i is not ready at or before $t_c = t_h - (q+3)$, T_i cannot be scheduled before $t_h - 1$ in $\mathcal{S}'$.

We next show that if T_i's predecessor T_j exists and is scheduled at $t_h - 1$ in $\mathcal{S}$, then it is not scheduled earlier in $\mathcal{S}'$. Because T_i is scheduled at t_h and T is in $A_0(t_h)$, T_i's tardiness is zero, and hence, by Lemma 16, $\mathsf{d}(T_i) = t_h + 1$. Hence, $\mathsf{d}(T_j) \leq t_h$ holds. If $\mathsf{d}(T_j) < t_h$ holds, then, by Claim 4, T_j is not displaced. In the other case, namely, $\mathsf{d}(T_j) = t_h$, by Lemma 2, $\mathsf{r}(T_i) \geq t_h - 1$, and hence, $|\omega(T_i)| = \mathsf{d}(T_i) - \mathsf{r}(T_i) \leq (t_h+1) - (t_h-1) = 2$ holds. Therefore, by Lemma 1, $|\omega(T_j)| \leq 3$, and hence, $\mathsf{r}(T_j) \geq \mathsf{d}(T_j) - 3 = t_h - 3$. If T_j is scheduled at $t_h - 1$ in $\mathcal{S}$, then by Lemma 7(a), $\mathsf{e}(T_j) \geq t_h - 3$. However, because $q \geq 1$, by (C), the deadline of every subtask scheduled in $[t_h - 3, t_h - 1)$ is at or before $t_h - q$, and hence, by Claim 4, no such subtask is displaced or removed. Therefore, in this case too, if T_j is scheduled at $t_h - 1$ in $\mathcal{S}$, it is not scheduled earlier in $\mathcal{S}'$. Thus, no subtask of T that is scheduled at or after $t_h - 1$ in $\mathcal{S}$ is scheduled before $t_h - 1$ in $\mathcal{S}'$.

We are now ready to establish the lag of T at $t_h - 1$ in $\mathcal{S}'$. By (11), we have

$$
\begin{aligned}
&\mathsf{lag}(\tau', t_h - 1, \mathcal{S}') \\
&= \mathsf{A}(\mathsf{ideal}_{\tau'}, T, 0, t_h - 1) - \mathsf{A}(\mathcal{S}', T, 0, t_h - 1) \\
&= \mathsf{A}(\mathsf{ideal}_{\tau}, T, 0, t_h + 1) - \mathsf{A}(\mathsf{ideal}_{\tau}, T, t_h - 1, t_h + 1) \\
&\quad -(\mathsf{A}(\mathcal{S}, T, 0, t_h + 1) - \mathsf{A}(\mathcal{S}, T, t_h - 1, t_h + 1)) \\
&\quad \text{(because, by (R), no subtask of } T \text{ is removed, and no subtask of } T \text{ scheduled} \\
&\quad \text{at or after } t_h - 1 \text{ in } \mathcal{S} \text{ is scheduled before } t_h - 1 \text{ in } \mathcal{S}') \\
&\geq \mathsf{A}(\mathsf{ideal}_{\tau}, T, 0, t_h + 1) - 2 \cdot W_{\max} - (\mathsf{A}(\mathcal{S}, T, 0, t_h + 1) - \mathsf{A}(\mathcal{S}, T, t_h - 1, t_h + 1)) \\
&\quad \text{(by (7) and (29))} \\
&\geq \mathsf{A}(\mathsf{ideal}_{\tau}, T, 0, t_h + 1) - 2 \cdot W_{\max} - \mathsf{A}(\mathcal{S}, T, 0, t_h + 1) + 1 \\
&\quad \text{(because at least subtask } T_i \text{ of } T \text{ is scheduled in } [t_h - 1, t_h + 1) \text{ in } \mathcal{S}) \\
&= \mathsf{lag}(T, t_h + 1, \mathcal{S}) - 2 \cdot W_{\max} + 1. \qquad \blacksquare
\end{aligned}
$$

Lemma 32 *Let T be a task with a subtask in τ_s^5. Then,* $\mathsf{lag}(T, t_h - 1, \mathcal{S}') \geq \mathsf{lag}(T, t_h + 1, \mathcal{S}) + 2 - 2 \cdot W_{\max}$.

Proof: As with Lemma 31, we first show that no subtask of T is removed. Because T is in τ^5, by Lemma 28(d), T is not in τ^i, where $1 \leq i \leq 6$ and $i \neq 5$. Hence, by Lemma 28(c), T is also not in τ^7 or in τ^8. Thus, T does not have a subtask in τ_s^R, and hence, no subtask of T is removed.

We next show that the subtasks of T scheduled at t_h or $t_h - 1$ are not displaced.

Let T_i be T's subtask scheduled at t_h. By the definition of A_q and A_{q-1}, the tardiness of T_i is greater than zero, and hence, $\mathsf{d}(T_i) \leq t_h$. Let T_j be T_i's predecessor. By the definition of τ_s^5, T_j exists. Further, $\mathsf{d}(T_j) \leq t_h - 1$ holds and T_j is scheduled at $t_h - 1$.

We now show that T_i and T_j are not displaced. For this, observe that because $\mathsf{d}(T_j) \leq t_h - 1$ holds, T_j is not displaced by Claim 4. Therefore, because T_i is T_j's successor, T_i is not ready to be scheduled until t_h, and hence, is not displaced either.

The above facts can be used to determine the lag of T at $t_h - 1$ in $\mathcal{S}'$ as follows. By (11), we have

$$\begin{aligned}
&\mathsf{lag}(\tau', t_h - 1, \mathcal{S}') \\
&= \mathsf{A}(\mathsf{ideal}_{\tau'}, T, 0, t_h - 1) - \mathsf{A}(\mathcal{S}', T, 0, t_h - 1) \\
&= \mathsf{A}(\mathsf{ideal}_{\tau}, T, 0, t_h + 1) - \mathsf{A}(\mathsf{ideal}_{\tau}, T, t_h - 1, t_h + 1) \\
&\quad -(\mathsf{A}(\mathcal{S}, T, 0, t_h + 1) - \mathsf{A}(\mathcal{S}, T, t_h - 1, t_h + 1)) \\
&\quad \text{(because no subtask of } T \text{ is removed, and because neither } T_i \text{ nor } T_j \text{ is displaced,} \\
&\quad \text{no subtask of } T \text{ scheduled at or after } t_h - 1 \text{ in } \mathcal{S} \text{ is scheduled before } t_h - 1 \text{ in } \mathcal{S}') \\
&\geq \mathsf{A}(\mathsf{ideal}_{\tau}, T, 0, t_h + 1) - 2 \cdot W_{\max} - (\mathsf{A}(\mathcal{S}, T, 0, t_h + 1) - \mathsf{A}(\mathcal{S}, T, t_h - 1, t_h + 1)) \\
&\quad \text{(by (7) and (29))} \\
&= \mathsf{A}(\mathsf{ideal}_{\tau}, T, 0, t_h + 1) - 2 \cdot W_{\max} - \mathsf{A}(\mathcal{S}, T, 0, t_h + 1) + 2 \\
&\quad \text{(because exactly two subtasks of } T\text{, } T_i \text{ and } T_j\text{, are scheduled in } [t_h - 1, t_h + 1)) \\
&= \mathsf{lag}(T, t_h + 1, \mathcal{S}) - 2 \cdot W_{\max} + 2 \qquad \blacksquare
\end{aligned}$$

Lemma 33 *Let T be a task with a subtask in τ_s^6. Then,* $\mathsf{lag}(T, t_h - 1, \mathcal{S}') > \mathsf{lag}(T, t_h + 1, \mathcal{S})$.

Proof: Let T_i denote T's subtask in τ_s^6. Because there is a hole in t_h (by (H)) and T is not scheduled at t_h, the eligibility time, and hence, the release time of T_i's successor is at least $t_h + 1$. However, by Claim 5, the release time of every subtask in τ is at most t_h. Therefore, T_i does not have a successor.

Since T_i is not in τ_s^2, $\mathsf{d}(T_i) \leq t_h - 1$ holds. Thus, all subtasks of T have their deadlines by $t_h - 1$ and complete executing by t_h in both ideal_τ and $\mathcal{S}$. Therefore, T's lag at $t_h + 1$ in $\mathcal{S}$ is zero.

Because $\mathsf{d}(T_i) \leq t_h - 1$ and T_i does not have a successor, by Claim 4, no subtask of T is displaced. Thus, in the ideal schedule for τ', subtasks of T complete executing by $t_h - 1$, whereas T_i is not complete until t_h in $\mathcal{S}'$. Thus, $\mathsf{lag}(T, t_h - 1, \mathcal{S}') > 0$, from which the lemma follows. ■

Lemma 34 *Let T be a task in τ^c. Then,* $\mathsf{lag}(T, t_h - 1, \mathcal{S}') = \mathsf{lag}(T, t_h + 1, \mathcal{S})$.

Proof: Because T is in τ^c, T does not contain a subtask in sets τ_s^i, where $1 \leq i \leq 8$. Hence, T does not have a subtask that is removed. We next show that T does not have a subtask that is scheduled at t_h or $t_h - 1$.

If T has a subtask T_i that is scheduled at t_h, then T is in A. By the condition of this case (Case C), $A_q^0 = \emptyset$ and $A_{q-1}^0 = \emptyset$. Hence, by (25), T is in one of $A_0(t_h)$, $A_q^1(t_h)$, $A_q^2(t_h)$, and $A_{q-1}^i(t_h)$, where $i \geq 1$. However, if T is in $A_0(t_h)$, then T_i is in τ_s^3 or τ_s^4. On the other hand, if T is in one of the remaining sets, then T_i has a tardiness greater than zero, but is not a c-MI, and hence, T is scheduled at $t_h - 1$; therefore, T_i is in τ_s^5. Thus, T_i is in one of τ_s^3, τ_s^4, and τ_s^5, and hence, T is in one of τ^3, τ^4, and τ^5. This contradicts the fact that T is in τ^c. Therefore, T cannot have a subtask scheduled at t_h.

We now show that T does not have a subtask scheduled at $t_h - 1$. By the definitions of τ_s^2 and τ_s^6, any subtask that is scheduled at $t_h - 1$, but does not have a later subtask of its task scheduled at t_h, is in one of these two subsets. Therefore, if T has a subtask T_i scheduled at $t_h - 1$, then because T is in τ^c (and hence not in τ^2 or τ^6), T is scheduled at t_h also. But as was shown above, T is not scheduled at t_h, and hence, is not scheduled at $t_h - 1$ either. Thus, T is not scheduled in either t_h or $t_h - 1$.

By Claim 5, no subtask of T is released at or after t_h+1. Therefore, because there is a hole in t_h, and T is not scheduled in either t_h or t_h-1, every subtask of T is scheduled before t_h-1, and completes executing by t_h-1 in $\mathcal{S}$. Hence, because there is a hole in t_h, by Lemma 15, the deadline of every subtask of T is at or before t_h+1.

To complete the proof, we show that the deadline of every subtask of T is at most t_h-1. Suppose to the contrary some subtask of T has its deadline after t_h-1. Let T_i be such a subtask with the largest index. Then, T_i is the critical subtask of T at either t_h or t_h-1 or at both times. Because T is not scheduled at either t_h or t_h-1, T_i is scheduled before t_h-1. Hence, T is in $B(t_h-1)$ or $B(t_h)$ or both. Also, because $\mathsf{d}(T_i) \geq t_h$ holds, by Definition 9, t'_b exists and T is scheduled at or before t'_b. But then, by the definition of τ_s^1, T_i is in τ_s^1, which contradicts the fact that T_i is in τ^c. Therefore, our assumption that T has a subtask with deadline after t_h-1 is incorrect.

Thus, all subtasks of T complete executing by t_h-1 in both the ideal schedules. Hence, the lag of T in $\mathcal{S}$ at t_h+1 is zero.

Because no subtask of T is removed or displaced, and every subtask of T is scheduled before t_h-1 in $\mathcal{S}$, all subtasks of T complete executing by t_h-1 in $\mathcal{S}'$ also. Therefore, $\mathsf{lag}(T, t_h-1, \mathcal{S}') = 0$. The lemma follows. ■

Lemma 36 *The roots of* $f(W_{\max}) = 2(M-h)(q+1)W_{\max}^2 - (q+2)(M-h)W_{\max} - ((q-1)M+1+h) = 0$ *are* $W_{\max} = \frac{(q+2)(M-h)\pm\sqrt{9q^2(M-h)^2+\Delta}}{4(M-h)(q+1)}$, *where* $\Delta = 4(M-h)(M(q-1)+h(2q^2+2q+1)+2q+2)$.

Proof: The roots of $f(W_{\max})$ are given by $\frac{(q+2)(M-h)\pm\sqrt{(q+2)^2(M-h)^2+8(M-h)(q+1)((q-1)M+1+h)}}{4(M-h)(q+1)}$. Let $I = (q+2)^2(M-h)^2 + 8(M-h)(q+1)((q-1)M+1+h)$ (the term within the square root). Then,

$$\begin{aligned}
I &= (q+2)^2(M-h)^2 + 8(M-h)(q+1)((q-1)M+1+h) \\
&= q^2(M-h)^2 + (4q+4)(M-h)^2 + 8q^2(M-h)^2 - 8q^2(M-h)^2 \\
&\quad +8(M-h)(q+1)((q-1)M+1+h) \\
&\qquad\qquad \text{(splitting the first term, and adding and subtracting } 8q^2(M-h)^2\text{)} \\
&= 9q^2(M-h)^2 + 4(M-h)(M(q-1)+h(2q^2+q+1)+2q+2) \\
&= 9q^2(M-h)^2 + \Delta.
\end{aligned}$$

■