A Unified Hard/Soft Real-Time Schedulability Test for Global EDF Multiprocessor Scheduling*

Hennadiy Leontyev and James H. Anderson
Department of Computer Science, The University of North Carolina at Chapel Hill

Abstract

The issue of deadline tardiness is considered under global earliest-deadline-first (GEDF) *multiprocessor scheduling. New schedulability tests are presented for determining whether a set of sporadic tasks with arbitrary relative deadlines can be scheduled under either preemptive or non-preemptive* GEDF *so that pre-defined tardiness bounds are met. These tests are of pseudo-polynomial time complexity, and can be used in hard real-time, soft real-time, and mixed contexts.*

1. Introduction

Most major chip manufacturers are investing in multicore technologies to continue performance improvements in their product lines in the face of fundamental limitations of single-core chip designs. This development is profound as it means that multiprocessors are now a “common-case” platform. This realization has led to renewed recent interest in multiprocessor real-time scheduling.

In research on this topic, much work has been directed at the sporadic task model, wherein tasks repeatedly generate sequential *jobs* subject to deadlines. A sporadic task has a specific *period*, which defines the minimum spacing between its jobs, and a *relative deadline*, which defines the length of the time interval in which each of its jobs is allowed to complete. Job deadlines can be either *hard* — in which case they should always be met — or *soft* — in which case misses can occur, provided the extent of violation is constrained in some way. In this paper, we consider a unified framework in which all timing constraints are specified using *deadline tardiness bounds*. For a hard deadline, tardiness must be zero, while for a soft deadline, it may be non-zero. We consider a real-time system to be *schedulable* using some scheduling algorithm A if no deadline tardiness bound is exceeded under A.

In work on multiprocessor scheduling, two basic scheduling approaches have been considered: *partitioning* and *global scheduling*. Under partitioning, tasks are statically assigned to processors, and a uniprocessor scheduling algorithm is used on each processor to schedule its assigned tasks. In contrast, under global scheduling, a task may execute on any processor and may migrate among processors. Global scheduling becomes a more viable option on multicore platforms due to the presence of on-chip shared caches, which lessen task migration overheads. One global algorithm that has received considerable attention is the *global earliest-deadline-first* (GEDF) algorithm. Under it, jobs are are globally prioritized in order of (absolute) deadline.

Motivation. Many systems have both hard and soft real-time components. One way to ensure task timing constraints in such a system is to treat all deadlines as hard. Perhaps partly because of that, most prior work on GEDF has focused on hard real-time schedulability tests [1, 4, 5, 6]. If such a test passes, then each task is guaranteed zero tardiness. Unfortunately, ensuring zero tardiness under GEDF may restrict system utilization severely. Such restrictions can be eased by allowing deadlines to be missed. For example, if only bounded tardiness is required (for *some* bound), then as shown by Devi and Anderson [9], restrictive utilization caps are not required under preemptive and non-preemptive GEDF. Unfortunately, as discussed below, the analysis in [9] imposes several restrictions that preclude the consideration of arbitrary task systems with *specified* tardiness bounds. Motivated by these concerns, in this paper, we present new schedulability tests for the non-preemptive and preemptive GEDF scheduling algorithms, henceforth denoted NP-GEDF and GEDF, respectively, that ensure that arbitrary pre-defined deadline tardiness bounds (including bounds of zero) are met.

Prior work. Of the several existing hard real-time schedulability tests for GEDF, the most accurate and sophisticated is due to Baruah [4]. This test requires pseudo-polynomial time complexity. It is also limited to *constrained deadline* systems, wherein each task’s relative deadline is at most its period. In recent work, Baruah and Baker presented a pseudo-polynomial test that lifts the latter restriction [5]. However, this test is rather pessimistic, as we demonstrate later. For NP-GEDF, a hard real-time test has also been proposed by Baruah in [3], but it is limited to *implicit deadline* systems, wherein each task’s relative deadline equals its period. Also, the maximum worst-case task execution time is required to be less than the minimum task period.

In [9], Devi and Anderson presented closed-form expres-

*Work supported by grants from IBM and Intel Corps., by NSF grants CCF 0541056 and CNS 0615197, and by ARO grant W911NF-06-1-0425.

sions that bound maximum deadline tardiness under GEDF and NP-GEDF. In [10], they also presented a variant of GEDF that can ensure arbitrary deadline tardiness (including zero) for up to m selected privileged tasks, where m is the number of processors; other tasks are guaranteed bounded deadline tardiness. However, the algorithm cannot ensure arbitrary deadline tardiness bounds for more than m tasks. In both [9] and [10], only implicit-deadline systems are considered and each task's maximum tardiness is assumed to be at least its worst-case execution time. Moreover, it is not possible to freely *specify* desired tardiness bounds, so systems where different tardiness thresholds are specified for different classes of tasks cannot be analyzed.

To summarize, using the results discussed above, and given a task set with specified deadline tardiness bounds, there are two basic options: check that maximum deadline tardiness is zero using a hard real-time schedulability test; or check whether specified tardiness bounds exceed the bounds computed in [8, 9, 10]. Additionally, it is possible to modify each task by increasing its relative deadline by an amount up to its tardiness threshold. However, this changes how jobs are prioritized and may cause a task's actual (unmodified) deadlines to be missed more frequently. Each of these approaches has limitations in systems where tasks may be subject to various (possibly quite different) tardiness thresholds (some of which may be zero).

Contributions. In this paper, we present a unified framework for checking that arbitrary pre-defined tardiness bounds are not violated under GEDF and NP-GEDF. Our work differs from prior work in several ways. First, in contrast to [4, 5], we consider non-preemptive job execution and non-zero tardiness bounds; in contrast to [4], we also consider arbitrary relative deadlines. Second, unlike in [8, 9, 10], we allow relative deadlines to differ from task periods and tardiness bounds to be arbitrary. Fourth, in contrast to [3], the presented test for NP-GEDF can be used to check hard real-time schedulability without severely restricting task parameters. Fifth, it is possible to apply our tests in settings where the *relative priorities* of jobs are specified using deadlines and *timeliness requirements* are specified using tardiness thresholds.

The rest of this paper is organized as follows. In Sec. 2, we present our task model. Then, in Sec. 3, we present the aforementioned GEDF and NP-GEDF schedulability tests. In Sec. 4, we experimentally compare them with other methods. In these experiments, our tests exhibited superior performance, typically by a wide margin. We conclude in Sec. 5.

2. System Model

We consider the problem of scheduling on m processors a set τ of n sporadic tasks, $T_1, \ldots, T_n$. Each task is invoked or *released* repeatedly, with each such invocation called a *job*. Associated with each task T_i are two parameters, e_i and p_i: e_i gives the worst-case *execution time* of one job of T_i, while, p_i, called the *period* of T_i, gives the minimum time between consecutive job releases of T_i.

The j^{th} job of T_i, where $j \geq 1$, is denoted $T_{i,j}$. A task's first job may be released at any time $t \geq 0$. The release time of job $T_{i,j}$ is denoted $r_{i,j}$ and its (absolute) deadline as $d_{i,j} = r_{i,j} + D_i$, where $D_i \geq e_i$ is the *relative deadline* of T_i. If the job $T_{i,j}$ completes at time t, then its *tardiness* is $\max(0, t - d_{i,j})$. A *task*'s tardiness is the maximum of the tardiness of any of its jobs. We denote the maximum allowed deadline tardiness for task T_i as $\Theta_i \geq 0$. We say that an unfinished job is *ready* if it has been released and its predecessor (if any) has completed execution.[1]

Throughout the paper, we assume that e_i, p_i, D_i, and Θ_i are non-negative integers and all time values are integral. For brevity, we sometimes use the notation $T_i(e_i, p_i, D_i, \Theta_i)$ to specify task parameters.

The *utilization of task* T_i is defined as $u_i = e_i/P_i$, and the *utilization of the task system* τ as $U_{sum} = \sum_{T_i \in \tau} u_i$. We assume $U_{sum} \leq m$, for otherwise, tardiness may grow unboundedly.

In what follows, we assume that each job $T_{a,b}$ has a unique priority so that $T_{a,b}$ has higher priority than $T_{k,j}$, denoted $T_{a,b} \prec T_{k,j}$, iff $d_{a,b} < d_{k,j} \vee (d_{a,b} = d_{k,j} \wedge b < j)$. $T_{a,b} \preceq T_{k,j}$ implies that either $T_{a,b} \prec T_{k,j}$ or $T_{a,b} = T_{k,j}$. GEDF selects at most m ready jobs with the highest priority so that lower-priority jobs can be preempted. In contrast, under NP-GEDF, all jobs execute non-preemptively.

In the next section, we devise GEDF and NP-GEDF tests that, when given a sporadic task set τ with arbitrary relative deadlines as described above, determine whether task T_i's maximum deadline tardiness is at most Θ_i. Due to space constraints, a full proof is given only for NP-GEDF, as it is more interesting. GEDF is dealt with by noting simplifications to the presented analysis that follow when preemptivity is allowed.

3. Schedulability Tests

In devising our tests, we adapt an approach due to Baruah for checking hard real-time schedulability under GEDF [4]. Adaptations are needed to deal with non-preemptivity (as allowed by NP-GEDF) and non-zero tardiness.

Similarly to [4], we order jobs by their priorities and assume that $T_{k,j}$ is the first job to miss its deadline, at time $t_d = d_{k,j}$, by more than its pre-defined tardiness threshold, Θ_k. We further assume that each job $T_{a,b}$ such that $T_{a,b} \prec T_{k,j}$ misses its deadline by at most Θ_a. We consider an interval that includes the time when $T_{k,j}$ becomes

[1]Note that, when a job of a task misses its deadline, the release time of the next job of that task is not altered. However, at most one job of a task may execute at any time, even if deadlines are missed.

ready and the latest time where $T_{k,j}$ is allowed to complete. During this interval, we consider demand due to competing higher-priority jobs and that due to lower-priority (non-preemptive) jobs that can interfere with $T_{k,j}$. We first compute a lower bound $LB(\tau, m)$ on this demand that is necessary for $T_{k,j}$'s tardiness to exceed Θ_k We then find a finite upper bound $UB(\tau, m)$ on this competing demand. Setting $UB(\tau, m) \geq LB(\tau, m)$ will give us a sufficient test for checking whether a task's tardiness bound is violated.

Definition 1. We let t_e denote the time when job $T_{k,j}$ becomes ready, which is the later of its arrival time $r_{k,j}$ and the time when $T_{k,j-1}$ (if $j \geq 2$) finishes execution.

Because $T_{k,j}$ misses its deadline, there are other jobs that deprive it of processor time after time t_e. These could be higher-priority jobs or lower-priority jobs that execute non-preemptively at time t_e.

Definition 2. We say that job $T_{i,j}$ is *pending* at time t if $r_{i,j} \leq t$ and $T_{i,j}$ has not completed execution at time t. We let $\tau_p(t) = \{T_a \mid \text{for some } b,$ $T_{a,b}$ is pending at time t and $T_{a,b} \preceq T_{k,j}\}$.

The following lemma identifies conditions under which $T_{k,j}$'s tardiness exceeds Θ_k.

Lemma 1. *If $t_e > t_d - \min(D_i)$ and $T_{k,j}$'s tardiness exceeds Θ_k, then $|\tau_p(t_e)| \geq m$ or fewer than $|\tau_p(t_e)|$ tasks from $\tau_p(t_e)$ execute at t_e.*

Proof. Suppose that $|\tau_p(t_e)| \leq m-1$ and all tasks in $\tau_p(t_e)$ execute at t_e. Job $T_{k,j}$'s release time is $r_{k,j} = t_d - D_k$. $T_{k,j}$'s predecessor completes at time $t' \leq t_d - p_k + \Theta_k$. Thus, by Def. 1,

$$t_e = \max(r_{k,j}, t') \leq \max(r_{k,j}, t_d - p_k + \Theta_k). \quad (1)$$

Because $|\tau_p(t_e)| \leq m-1$ and all tasks (including T_k) in $\tau_p(t_e)$ execute at t_e, job $T_{k,j}$ commences execution at t_e, and by (1), completes at or before $t_e + e_k \leq \max(r_{k,j} + e_k, t_d - p_k + \Theta_k + e_k) \leq \max(r_{k,j} + D_k, t_d + \Theta_k) \leq \max(t_d, t_d + \Theta_k) \leq t_d + \Theta_k$. This contradicts the assumption that $T_{k,j}$'s tardiness exceeds Θ_k.[2] □

Definition 3. Let $t_0 \leq t_e$ be the earliest instant such that $\forall t \in [t_0, t_e)$, $|\tau_p(t)| \geq m$ or fewer than $|\tau_p(t)|$ tasks from $\tau_p(t)$ execute at time t. If such an instant does not exist, then let $t_0 \triangleq t_e$.

[2]Note that the condition $t_e > t_d - \min(D_i)$ is not used in the above argument. This condition has been included so that the lemma is applicable to GEDF as well. In this case, no job $T_{a,b}$ such that $T_{a,b} \preceq T_{k,j}$ can be released at or after t_e and preempt $T_{k,j}$. The argument used for NP-GEDF can thus be repeated.

Def. 3 generalizes the well-known concept of an *idle instant* in uniprocessor scheduling. We call an interval $[t_1, t_2)$ *busy* if no processor is idle within it.

Claim 1. *The time interval $[t_0, t_e)$ is busy.*

Proof. Suppose that a processor is idle at time $t \in [t_0, t_e)$. Because NP-GEDF is work-conserving, all tasks in $\tau_p(t)$ execute at time t and thus $|\tau_p(t)| \leq m-1$, which violates Def. 3. □

If job $T_{k,j}$ executes for $x_{k,j}$ time units within the interval $[t_e, t_d + \Theta_k)$, where $x_{k,j} \leq e_k$ is the actual execution time of $T_{k,j}$, then it cannot violate its tardiness threshold. If job $T_{k,j}$ executes for less than $x_{k,j}$ time units within $[t_e, t_d + \Theta_k)$, then it executes for at most $x_{k,j} - 1$ time units within this interval, as time is integral. Hence, if $T_{k,j}$ misses its deadline by more than Θ_k, then the total time for which it *does not* execute in $[t_e, t_d + \Theta_k)$ is at least $t_d + \Theta_k - t_e - (x_{k,j} - 1) \geq t_d + \Theta_k - t_e - (e_k - 1) = t_d + \Theta_k - t_e - e_k + 1$.

Definition 4. Let Γ be a subset of the set of intervals within $[t_e, t_d + \Theta_k)$, where job $T_{k,j}$ does not execute, such that the cumulative length of Γ is *exactly* $t_e - t_d + \Theta_k - e_k + 1$.

Example 1. Fig. 1 (a) shows a schedule where job $T_{k,j}$, released at time $r_{k,j}$, misses its deadline at time t_d by more than Θ_k time units under NP-GEDF. The schedule shows the demand placed on all m processors. (It is not intended to depict a particular assignment of jobs to processors.) The predecessor job $T_{k,j-1}$ misses its deadline at time $t_d - p_k$ by $B_k \leq \Theta_k$ time units, so that job $T_{k,j}$ becomes ready at time t_e. At time $t_0 \leq t_e$, there are q tasks with non-preemptive lower-priority jobs that execute and at least $m - q + 1$ tasks with pending higher-priority jobs. Because at least one of these tasks is not scheduled at t_0, the q non-preemptive jobs must have commenced execution before t_0. This implies that at most $m - q$ tasks have pending higher-priority jobs executing immediately prior to t_0. Job $T_{k,j}$ does not commence execution immediately at time t_e because other jobs execute during Γ. Note that, in Fig. 1 (a), Γ consists of one interval because $T_{k,j}$ cannot be preempted once it starts execution. The situation is different in Fig. 1 (b), which shows a GEDF schedule that we examine later. □

In Fig. 1, the area of the lightly-shaded intervals corresponds to the competing demand for the job $T_{k,j}$. We now derive a condition this demand must satisfy if the tardiness threshold for T_k is violated.

Definition 5. Let $I(T_i)$ be the total amount of time for which jobs of task T_i execute within $[t_0, t_e) \cup \Gamma$.

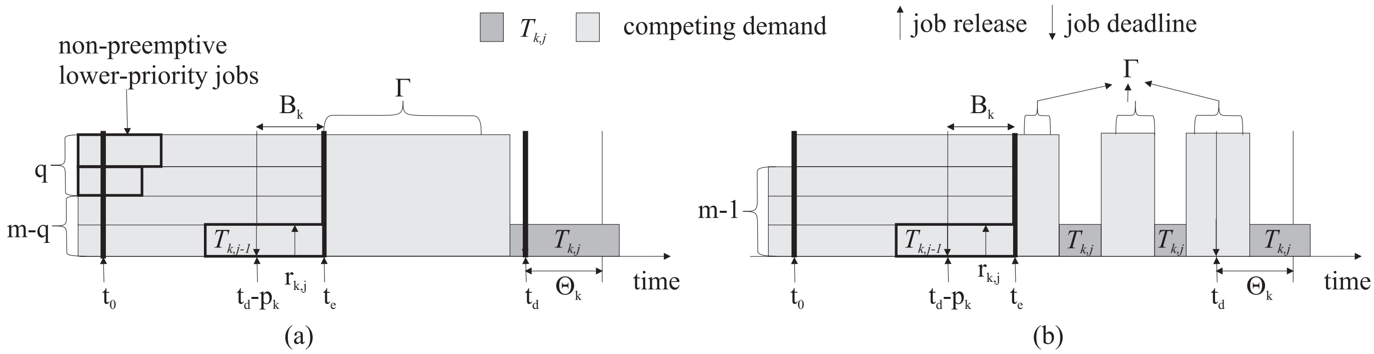

Figure 1. Conditions for tardiness violations under (a) NP-GEDF **and (b)** GEDF**.**

If job $T_{k,j}$ violates its tardiness threshold Θ_k, then

$$\sum_{i=1}^{n} I(T_i) \geq m \cdot (t_d - t_0 + \Theta_k - e_k + 1). \tag{2}$$

This follows from two observations. First, since $T_{k,j}$ does not execute within Γ while being ready, all processors execute jobs of tasks other than T_k during Γ. Second, all processors are busy during $[t_0, t_e)$, by Claim 1. Multiplying the total length of $[t_0, t_e) \cup \Gamma$, which is $|[t_0, t_e) \cup \Gamma| = (t_e - t_0) + (t_d + \Theta_k - t_e - e_k + 1) = t_d - t_0 + \Theta_k - e_k + 1$, by m, we get the RHS of (2).

We construct a schedulability test using Inequality (2) as follows. In the following subsection, we derive a lower bound for the term $t_d - t_0$ in the RHS of (2). Then, in Sec. 3.2, we derive an upper bound for $I(T_i)$ in the LHS of (2). Later, we will show that these bounds can be used to obtain a schedulability test. To avoid distracting "boundary cases," we henceforth assume that the schedule being analyzed is prepended with a schedule in which no deadlines are missed that is long enough to ensure that $t_0 > 0$ holds and all predecessor jobs referenced in the proof exist.

3.1. Estimating $t_d - t_0$

Lemma 2. $t_d - t_0 \geq \max(\min(D_i), \min(D_k, p_k - \Theta_k))$.

Proof. By Def. 1 and (1),

$$\begin{aligned} t_0 &\leq t_e \\ &= \max(r_{k,j}, t_d - p_k + \Theta_k) \\ &\leq \max(t_d - D_k, t_d - p_k + \Theta_k). \end{aligned} \tag{3}$$

From this, we have,

$$\begin{aligned} -t_0 &\geq \min(-t_d + D_k, -t_d + p_k - \Theta_k) \\ \Rightarrow \quad t_d - t_0 &\geq \min(t_d - t_d + D_k, t_d - t_d + p_k - \Theta_k) \\ \Rightarrow \quad t_d - t_0 &\geq \min(D_k, p_k - \Theta_k). \end{aligned}$$

We next prove $t_d - t_0 \geq \min(D_i)$. Assume to the contrary that $t_0 > t_d - \min(D_i)$ holds. Then, $t_0 > t_d - D_k \geq 0$. Let $\gamma = \tau_p(t_0) \setminus \tau_p(t_0 - 1)$. By Def. 3, $|\tau_p(t_0 - 1)| \leq m - 1$ and all tasks in $\tau_p(t_0 - 1)$ execute at time $t_0 - 1$. Considering γ, there are two possibilities. If $\gamma = \emptyset$, then $\tau_p(t_0) \subseteq \tau_p(t_0 - 1)$ and hence $|\tau_p(t_0)| \leq m - 1$ and all tasks in $\tau_p(t_0)$ execute at time $t_0 - 1$. At time t_0, the scheduler cannot preempt a task in $\tau_p(t_0)$ and schedule a lower-priority task. Thus, all tasks in $\tau_p(t_0)$ execute at t_0, which violates Def. 3. On the other hand, if $\gamma \neq \emptyset$, then a task T_a that is not pending at $t_0 - 1$ releases a job $T_{a,b}$ at time $t_0 = r_{a,b}$ such that $d_{a,b} \leq t_d$. Since, by our assumption, $t_d - \min(D_i) < t_0 = r_{a,b}$, we have $d_{a,b} - \min(D_i) < r_{a,b}$ and hence $D_a = d_{a,b} - r_{a,b} < \min(D_i)$, a contradiction. □

3.2. Bounding $I(T_i)$

Definition 6. $\delta_k \triangleq t_d - t_0$.

From the above definition and (2),

$$\sum_{i=1}^{n} I(T_i) \geq m \cdot (\delta_k + \Theta_k - e_k + 1). \tag{4}$$

To check that pre-defined tardiness bounds are not violated we have to verify, for each task T_k, that $\sum_{i=1}^{n} I(T_i) < m \cdot (\delta_k + \Theta_k - e_k + 1)$ holds, where $\delta_k \geq \max(\min(D_i), \min(D_k, p_k - \Theta_k))$ (from Lemma 2). Because it is difficult to determine each $I(T_i)$ term exactly, we derive upper bounds for them.

The next lemma identifies jobs that cannot interfere with $T_{k,j}$. Its proof is straightforward and is therefore omitted.

Lemma 3. *Let $T_{a,b}$ be a job of task T_a, where $T_{a,b} \succ T_{k,j}$, such that $T_{a,b}$'s predecessor completes by time t_0, and either $r_{a,b} < t_0$ and $T_{a,b}$ is pending but not executing at t_0 or $r_{a,b} \geq t_0$. Then, T_a does not execute during $[t_0, t_e) \cup \Gamma$ and $I(T_a) = 0$.*

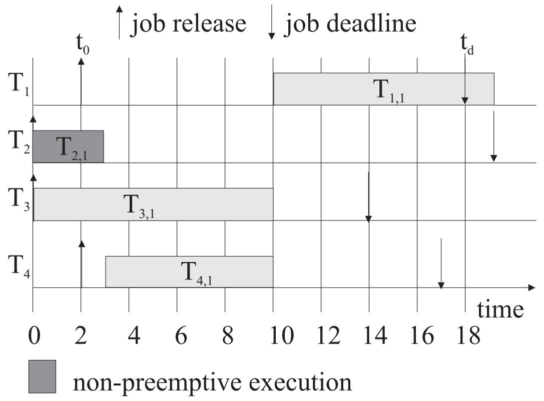

Figure 2. Classification of tasks into τ_{CH}, τ_{CL}, and τ_{NC}.

Using the lemma above, we can separate the tasks that may execute within $[t_0, t_e) \cup \Gamma$ into three disjoint sets (which are illustrated with an example below):

τ_{CH} **("high-priority carry-in"):** Includes each task T_a with a job $T_{a,b}$ that executes at time t_0 such that $T_{a,b} \preceq T_{k,j}$ and $r_{a,b} < t_0$.
τ_{CL} **("low-priority carry-in"):** Includes each task T_a with a job $T_{a,b}$ that executes at time t_0 such that $T_{a,b} \succ T_{k,j}$ and $r_{a,b} < t_0$.
τ_{NC} **("non-carry-in"):** Includes each task T_a with a job $T_{a,b}$, where $T_{a,b} \preceq T_{k,j}$, such that $T_{a,b}$ does not execute prior to t_0 and $T_{a,b}$'s predecessor completes by time t_0.

Note that a job of $T_a \in \tau_{CH}$ must be pending immediately prior to t_0, thus, by Def. 3, T_a must execute immediately prior to t_0. Also, a job of $T_a \in \tau_{CL}$ must be executing non-preemptively at time t_0.

Example 2. Consider the two-processor schedule in Fig. 2. In this schedule, job $T_{1,1}$, which corresponds to our job of interest $T_{k,j}$, becomes ready at time 2 and misses its deadline at time $t_d = d_{1,1} = 18$. At time 2, job $T_{2,1}$, which is released at time 0, executes non-preemptively and jobs $T_{3,1}$ and $T_{4,1}$ have higher priority than $T_{1,1}$. Note that $T_{4,1}$ becomes ready at time 2 and does not execute at that time. Also, $\tau_p(1) = \{T_3\}$ and T_3 executes at time 1, while $\tau_p(2) = \{T_3, T_4\}$ and only T_3 executes at time 2. Thus, $t_0 = 2$, $\tau_{CH} = \{T_3\}$, $\tau_{CL} = \{T_2\}$, and $\tau_{NC} = \{T_1, T_4\}$.

Claim 2. *If $\delta_k < D_k$, then $T_k \in \tau_{CH}$.*

Proof. By Def. 6, $t_0 = t_d - \delta_k$. Assuming $\delta_k < D_k$, this implies $t_0 > t_d - D_k = r_{k,j}$. Because $t_e \geq t_0 > r_{k,j}$, job $T_{k,j}$ is not ready at its release time, and one or more of $T_{k,j}$'s predecessors is pending throughout $[r_{k,j}, t_0)$. Let $T_{k,b}$, where $b < j$, be the earliest pending job of T_k at time $t_0 - 1$. Note that $T_{k,b} \prec T_{k,j}$. Thus, by the definition of $\tau_p(t_0 - 1)$ (see Def. 2), $T_k \in \tau_p(t_0 - 1)$, and hence, by Def. 3, $T_{k,b}$ executes at $t_0 - 1$. Since $r_{k,j} < t_0$, we have $r_{k,b} < t_0$, and thus, by the definition of τ_{CH}, $T_k \in \tau_{CH}$. □

We henceforth use $I_{CH}(T_i)$, $I_{CL}(T_i)$, and $I_{NC}(T_i)$ to denote $I(T_i)$ for the case when T_i is in τ_{CH}, τ_{CL}, and τ_{NC}, respectively. With this notation, (4) becomes

$$\sum_{T_i \in \tau_{CH}} I_{CH}(T_i) + \sum_{T_i \in \tau_{CL}} I_{CL}(T_i) + \sum_{T_i \in \tau_{NC}} I_{NC}(T_i) \geq m \cdot (\delta_k + \Theta_k - e_k + 1). \quad (5)$$

In order to verify that no tardiness bound is violated for any task in τ, we show that the negation of (5) holds for each task $T_k \in \tau$ by establishing the following:

$$\max\left(\sum_{T_i \in \tau_{CH}} I_{CH}(T_i) + \sum_{T_i \in \tau_{CL}} I_{CL}(T_i) + \sum_{T_i \in \tau_{NC}} I_{NC}(T_i)\right) < m \cdot (\delta_k + \Theta_k - e_k + 1). \quad (6)$$

This expression must be checked for each $\delta_k \geq \max(\min(D_i), \min(D_k, p_k - \Theta_k))$ (from Lemma 2), and each valid choice of τ_{CH}, τ_{CL}, and τ_{NC}. Due to the latter, we take the maximum over all possible choices in the LHS of (6).

We are left with bounding each of $I_{CH}(T_i)$, $I_{CL}(T_i)$, and $I_{NC}(T_i)$ and maximizing the total demand for any valid choice of the sets τ_{CH}, τ_{CL}, and τ_{NC}. Some trivial bounds can be derived easily.

Lemma 4. *Each of $I_{CH}(T_i)$, $I_{CL}(T_i)$, and $I_{NC}(T_i)$ is at most $\delta_k + \Theta_k - e_k + 1$, if $i \neq k$, and at most $\max(\delta_k - D_k, \delta_k - p_k + \Theta_k)$, if $i = k$.*

Proof. If $i \neq k$, then the work performed by T_i in $[t_0, t_e) \cup \Gamma$ cannot exceed the cumulative length of $[t_0, t_e) \cup \Gamma$, which is $\delta_k + \Theta_k - e_k + 1$, by Def. 4. Also, $I(T_k)$ cannot exceed the length of $[t_0, t_e)$ because T_k does not execute within Γ. By (3), we can bound $t_e - t_0$ as follows:

$$\begin{aligned} & t_e - t_0 \\ \leq \ & \max(t_d - D_k, t_d - p_k + \Theta_k) - t_0 \\ = \ & \max(t_d - t_0 - D_k, t_d - t_0 - p_k + \Theta_k) \\ & \{\text{by Def. 6}\} \\ \leq \ & \max(\delta_k - D_k, \delta_k - p_k + \Theta_k). \end{aligned}$$

□

Lemma 5. *$I_{CL}(T_i)$ is at most*

$$\begin{cases} \min(e_i - 1, \\ \quad \delta_k + \Theta_k - e_k + 1), & i \neq k \text{ and } D_i \geq \delta_k + 2 \\ & \text{or } i > k \text{ and } D_i \geq \delta_k + 1, \\ 0, & \text{otherwise.} \end{cases}$$

Proof. If $T_i \in \tau_{CL}$, then some job $T_{i,g}$, where $T_{i,g} \succ T_{k,j}$ and $r_{i,g} < t_0$, executes non-preemptively at t_0. Either $d_{i,g} > t_d$ or $d_{i,g} = t_d$ and $i > k$. If $d_{i,g} > t_d$, then T_i can be any task other than T_k for which $D_i \geq \delta_k + 2$ holds (since $\delta_k = t_d - t_0$). If $d_{i,g} = t_d$, then $D_i \geq \delta_k + 1$ and $i > k$. It follows from these facts that the only job of T_i that can execute in $[t_0, t_e) \cup \Gamma$ is $T_{i,g}$, and it can do so for at most $e_i - 1$ time units. Thus, $I_{CL}(T_i) \leq e_i - 1$. By Lemma 4, the required result follows. □

Lemma 6. *$I_{NC}(T_i)$ is at most*

$$\begin{cases} \min(DBF(T_i, \delta_k), \\ \quad \delta_k + \Theta_k - e_k + 1), & i \neq k \\ \min(DBF(T_i, \delta_k) - e_k, \\ \quad \max(\delta_k - D_k, \delta_k - p_k + \Theta_k)), & i = k \text{ and } \delta_k \geq D_k, \\ 0, & \text{otherwise,} \end{cases}$$

where

$$DBF(T_i, \delta_k) = \max(0, \left(\left\lfloor \frac{\delta_k - D_i}{p_i} \right\rfloor + 1 \right) \cdot e_i). \quad (7)$$

Proof. Consider $T_i \in \tau_{NC}$. There are two cases.
Case 1: $i \neq k$**.** Since $T_i \in \tau_{NC}$, each job $T_{i,g}$ that executes within $[t_0, t_e) \cup \Gamma$ has $r_{i,g} \geq t_0$ and $d_{i,g} \leq t_d$. According to [4], the demand due to such jobs is at most $DBF(T_i, t_d - t_0) = DBF(T_i, \delta_k)$, where DBF is given by (7). By Lemma 4, the required result follows.
Case 2: $i = k$**.** If $\delta_k < D_k$, then, by Claim 2, $T_k \notin \tau_{NC}$. If $\delta_k \geq D_k$, then the demand of jobs of T_k other than $T_{k,j}$ that are released and have deadlines within $[t_0, t_d]$ is at most $DBF(T_k, \delta_k) - e_k$. By Lemma 4, the required result follows. □

The following lemma is proved similarly to Lemma 6. Its proof can be found in an appendix.

Lemma 7. *$I_{CH}(T_i)$ is at most*

$$\begin{cases} \min(DBF'(T_i, \delta_k), \delta_k + \Theta_k - e_k + 1), & i \neq k, \\ \min(DBF'(T_i, \delta_k) - e_k, \\ \quad \max(\delta_k - D_k, \delta_k - p_k + \Theta_k)), & i = k, \end{cases}$$

where

$$DBF'(T_i, \delta_k) = \left\lfloor \frac{\delta_k + \Theta_i}{p_i} \right\rfloor \cdot e_i + \min(e_i, (\delta_k + \Theta_i) \bmod p_i)). \quad (8)$$

Each of $I_{CH}(T_i)$ and $I_{NC}(T_i)$ accounts for the demand of jobs of T_i (excluding that due to $T_{k,j}$ if $i = k$) having both release times and deadlines within the interval $[t_0, t_d]$. In addition, $I_{CH}(T_i)$ accounts for the demand of higher-priority jobs released prior to t_0 that are pending at t_0. The lemma below easily follows.

Lemma 8. *For each task T_i, $I_{CH}(T_i) \geq I_{NC}(T_i) \geq 0$ and $I_{CL}(T_i) \geq 0$.*

Let

$$M(T_k) = \sum_{T_i \in \tau_{CH}} I_{CH}(T_i) + \sum_{T_i \in \tau_{CL}} I_{CL}(T_i) + \sum_{T_i \in \tau_{NC}} I_{NC}(T_i). \quad (9)$$

To verify that (6) holds, we need to find the sets τ_{CH}, τ_{CL}, and τ_{NC} that would yield $M^*(T_k) = \max_{\tau_{CH}, \tau_{CL}, \tau_{NC}} M(T_k)$ for a given value δ_k. Not every choice of τ_{CH}, τ_{CL}, and τ_{NC} is valid. Overall, the sets τ_{CH}, τ_{CL}, and τ_{NC} must satisfy the following constraints:

$$\left. \begin{array}{ll} \tau_{NC} \cup \tau_{CH} \cup \tau_{CL} \subseteq \tau, & \tau_{NC} \cap \tau_{CH} \cap \tau_{CL} = \emptyset, \\ T_k \notin \tau_{CL}, & \delta_k < D_k \Rightarrow T_k \in \tau_{CH}, \\ |\tau_{CH}| \leq m - 1, & |\tau_{CH} \cup \tau_{CL}| \leq m. \end{array} \right\} \quad (10)$$

The constraint $T_k \notin \tau_{CL}$ follows because each task $T_i \in \tau_{CL}$ has a job executing at time t_0 of lower priority than $T_{k,j}$. The constraint $\delta_k < D_k \Rightarrow T_k \in \tau_{CH}$ follows from Claim 2. By Def. 2 and the definition of τ_{CH}, $\tau_{CH} \subseteq \tau_p(t_0 - 1)$. By Def. 3, all tasks in $\tau_p(t_0 - 1)$ execute at $t_0 - 1$ and $|\tau_p(t_0 - 1)| \leq m - 1$. Thus, $|\tau_{CH}| \leq m - 1$. Because at most m jobs may execute at any time, the number of tasks with carry-in jobs is $|\tau_{CH} \cup \tau_{CL}| \leq m$.

3.3. Finding $M^*(T_k)$

Given the constraints in (10), it is relatively straightforward to determine $M^*(T_k)$ in polynomial time. In this section, we briefly sketch how this can be done. In this description, we assume for simplicity that $n \geq m$ and $\delta_k \geq D_k$. The case $\delta_k < D_k$ only restricts T_k to be in τ_{CH} (by Claim 2) and can be dealt with similarly. Given the constraint $|\tau_{CL} \cup \tau_{CH}| \leq m$, we must merely check which of at most m tasks should be assigned to τ_{CH} and τ_{CL} and how such tasks should be distributed between these two sets. (By (10) and Lemma 8, we can assume that it is desirable to assign at least $m - 1$ tasks to $\tau_{CH} \cup \tau_{CL}$.) In addition, the constraint $|\tau_{CL}| \leq m - 1$ implies that at least one task is assigned to τ_{CL} if $|\tau_{CH} \cup \tau_{CL}| = m$.

The appropriate tasks to assign to $\tau_{CH} \cup \tau_{CL}$ can be determined as follows. To begin, imagine a default assignment wherein all tasks are assigned to τ_{NC}. Given such an assignment, we must select at most m tasks to move from τ_{NC} to

$\tau_{CH} \cup \tau_{CL}$ under the restrictions expressed above. For this purpose, define, for each task T_i, two values: $HD(T_i) = I_{CH}(T_i) - I_{NC}(T_i)$ and $LD(T_i) = I_{CL}(T_i) - I_{NC}(T_i)$. In these expressions, $I_{CH}(T_i)$, $I_{CL}(T_i)$, and $I_{NC}(T_i)$ are assumed to equal their upper bounds as given in Lemmas 5, 6, and 7. $HD(T_i)$ and $LD(T_i)$ reflect, respectively, how the value of the RHS of (9) will change if T_i is moved to τ_{CH} or τ_{CL}. We say that τ_{CL} (respectively, τ_{CH}) is T_i's *preferred group* if $LD(T_i) \geq HD(T_i)$ (respectively, $HD(T_i) > LD(T_i)$). Now, define $X(T_i) = \max(HD(T_i), LD(T_i))$. Assume that all tasks are ranked so that $i \leq j \Rightarrow X(T_i) \geq X(T_j)$. It is easy to show that there exists an optimal assignment (i.e., one that maximizes the RHS of (9)) in which **(i)** the top $m-1$ tasks in this ranking are assigned to $\tau_{CH} \cup \tau_{CL}$, and **(ii)** $m-2$ of these tasks are assigned to their preferred group. Given this, an optimal assignment can be found via a two-step process. First, consider each of the top $m-1$ ranked tasks in order and move each from τ_{NC} to its preferred group in $\tau_{CH} \cup \tau_{CL}$. Second, select an m^{th} task to move to $\tau_{CH} \cup \tau_{CL}$. If, after the first step, $|\tau_{CH}| < m-1$, then this can be done by simply moving the m^{th} task to its preferred group. However, if $|\tau_{CH}| = m-1$, then we must scan the remaining $n-m+1$ tasks, and record the value of the RHS of (9) that results when each such task is moved to each of τ_{CH} and τ_{CL}. When considering the possibility of moving such a task to τ_{CH} in this case, we must move some task currently assigned there to τ_{CL} (such a task is one of the top $m-1$ ranked tasks). All $m-1$ choices of a task to move to τ_{CL} must be considered. Note that these manipulations must respect the condition $T_k \notin \tau_{CL}$ (see (10)). After considering all possibilities in the process just described, we choose the assignment that maximizes the RHS of (9).

3.4. Schedulability Test

From the preceding discussion and analysis, we have the following theorem.

Theorem 1. *If*

$$M^*(T_k) < m \cdot (\delta_k + \Theta_k - e_k + 1), \tag{11}$$

holds for each sporadic task $T_k \in \tau$ *and* $\delta_k \geq \max(\min(D_i), \min(D_k, p_k - \Theta_k))$*, then no tardiness threshold in* τ *is violated under* NP-GEDF.

So far, we have analyzed only NP-GEDF. However, the reasoning requires only minor changes for GEDF. Under GEDF, lower-priority jobs can be excluded from consideration when analyzing job $T_{k,j}$'s behavior. In this case, time t_0 is the earliest time instant so that the interval $[t_0, t_e)$ is busy and one or more processors is idle before time t_0. In fact, this definition of t_0 is employed in [4]. Therefore, in Fig. 1 (b), which shows a GEDF schedule, the time interval $[t_0, t_e)$ is busy and at least one processor is idle immediately prior to t_0. Also, under GEDF, the set Γ may not be contiguous, as illustrated in Fig. 1 (b), because higher-priority jobs may preempt $T_{k,j}$. The set τ_{CL} is empty under GEDF, and hence, $I_{CL}(T_i) = 0$ for all tasks T_i.

If $\Theta_i = 0$, and $D_i \leq p_i$ for all tasks, then our GEDF test reduces to the hard real-time test in [4] (assuming integral time). Note that, since we have assumed that time is integral, our test's accuracy (particularly the LHS of (6)) depends on the assumed granularity of time.

3.5. Computational Complexity

In applying Theorem 1, we have to compute $M^*(T_k)$ and check (11) for an infinite number of values $\delta_k \geq \max(\min(D_i), \min(D_k, p_k - \Theta_k))$. However, the following theorem, proved in the appendix, bounds the range of δ_k.

Definition 7. Let $\delta_{max} = (E(m) + U(m-1) \cdot \max(\Theta_i) + R + m \cdot (e_k - \Theta_k - 1))/(m - U_{sum})$, where $E(y)$ (respectively, $U(y)$) is the sum of y largest task execution times (respectively, utilizations), and $R = \sum_{T_i \in \tau} \max(0, u_i \cdot (p_i - D_i))$.

Theorem 2. *If (11) holds for task* T_k *for all* $\delta_k \in [\max(\min(D_i), \min(D_k, p_k - \Theta_k)), \delta_{max}]$*, then (11)holds for all* $\delta_k \geq \max(\min(D_i), \min(D_k, p_k - \Theta_k))$.

It can also be shown that (11) only needs to be tested at those values of δ_k at which $DBF(T_i, \delta_k)$ changes for some T_i. Given that $M^*(T_k)$ can be determined in polynomial time, Theorem 2 implies the following

Corollary 1. *The condition in Theorem 1 can be tested in time pseudo-polynomial with respect to the task parameters, for all task systems* τ *for which* U_{sum} *is bounded by a constant strictly less than the number of processors* m.

4. Experiments

To evaluate the efficacy of our new tests, we compared them to other known hard and soft real-time schedulability tests for GEDF and NP-GEDF when checking randomly-generated task sets. In this section, we discuss the results of this evaluation.

Task generation procedure. In generating task systems, we adopted the methodology proposed by Baker in [2]. Integral task periods were distributed uniformly over $[1000, 100000]$. (In [2], the range [1,1000] was proposed. In essence, we are assuming integral time is defined at a finer granularity.) Integral task execution times were computed using periods and utilizations. Relative deadlines were defined to be either implicit (i.e., $D_i = p_i$ for each task T_i) or restricted, in which case D_i was uniformly distributed over $[e_i, p_i]$. Four utilization distributions were considered, truncated to the range $[0.001, 0.999]$:

U1: uniform over $[0.001, 0.999]$;
U2: bimodal: uniform over $[0.1, 0.5]$ with probability $2/3$ and over $[0.5, 1]$ with probability $1/3$;
U3: exponential with mean 0.25;
U4: exponential with mean 0.5.

Task sets were generated for $m = 2$, 4, and 8 processors, as follows. An initial set of $m+1$ tasks was generated and then tasks were iteratively added until total utilization exceeded m. The schedulability of each of these generated sets was checked using each tested scheme. After checking all such task sets, the entire procedure was repeated, starting with a new initial set of $m+1$ tasks.

Under NP-GEDF, we only considered the utilization distribution U1. We also slightly modified the task generation procedure to ensure that $\min(p_i) > \max(e_i)$ holds for each task set. This was done in order for the hard real-time test in [3] to be applicable to all generated task sets (see below).

We examined three rules for setting each task T_i's tardiness threshold Θ_i:

R1: $\Theta_i = \min(\alpha \cdot p_i, 5 \cdot p_i)$, where α has a Poisson distribution with mean 1;

R2: $\Theta_i = \begin{cases} 0, & \text{with probability } 0.2, \\ 0.5 \cdot p_i, & \text{with probability } 0.8; \end{cases}$

R3: $\Theta_i = \begin{cases} \textit{uniform}(0, p_i) & \text{if } p_i < 5000, \\ \textit{uniform}(p_i, 2 \cdot p_i), & \text{otherwise.} \end{cases}$

For Rule R1, approximately one third of all tasks have $\Theta_i = 0$, one third have $\Theta_i = p_i$, and one fifth have $\Theta_i = 2 \cdot p_i$. For Rule R2, approximately 20% of all tasks are hard real-time tasks. For Rule R3, tasks with short periods have low tardiness thresholds. For each combination of scheduling algorithm, utilization distribution, processor count, and tardiness rule, we generated $250,000$ task sets.

Schedulability tests. The schedulability of each generated task set $\tau = \{T_i(e_i, p_i, D_i, \Theta_i)\}$ was checked via four basic approaches:

• *View the system as hard real-time.* Under GEDF, we tested whether the task set $\tau^h = \{T_i^h(e_i, p_i, D_i, 0)\}$ (note $\Theta_i = 0$) is schedulable using the test in [4], denoted as SB. Under NP-GEDF, we checked the schedulability of τ^h using the test from [3], which is denoted NP.

• *Check previously-established tardiness bounds.* We checked whether the tardiness bound computed for each task $T_i \in \tau$ using the analysis from [8, 9] is at most Θ_i. These tests, denoted DA for GEDF and NP-DA for NP-GEDF, are applicable for the case of implicit deadlines.

• *Use the results of our paper.* We tested the schedulability of τ using Theorem 1. The resulting tests are denoted LA for GEDF and NP-LA for NP-GEDF.

• *Use extended task deadlines.* We tested the schedulability of the task set $\tau^x = \{T_i^x(e_i, p_i, D_i + \Theta_i, 0)\}$, where each task's relative deadline is extended by its respective tardiness threshold and its required tardiness is zero. For GEDF, the analysis from [4] cannot be applied to τ^x, because a task's relative deadline may exceed its period in τ^x. Thus, under GEDF we checked the schedulability of τ^x using the test from [5], which is denoted BB-ext. We also used Theorem 1 for both GEDF and NP-GEDF. These tests are denoted LA-ext and NP-LA-ext, respectively.

To constrain computation times, we deemed a task set to be unschedulable using SB and any of the above LA variants if $U_{sum} > m - 0.01$.

Results. Due to space constraints, we discuss results only for the case where $m = 4$, U1 is the utilization distribution, and all deadlines are implicit. However, other omitted cases showed similar trends. (Note that a greater range of schedulability approaches can be compared in the case of implicit deadlines, as several such approaches require $D_i = p_i$.) In Fig. 3 the total number of generated tasks is shown along with the number of task sets deemed schedulable by each applicable test as a function of total system utilization for the case under consideration. The three columns correspond to tardiness Rules R1, R2, and R3, respectively. The two rows correspond to GEDF and NP-GEDF, respectively.

Insets (a)–(c) show that LA and LA-ext are superior for GEDF due to the flexibility of Theorem 1 when analyzing individual task timing constraints. However, we observed that for task systems with low per-task utilizations (a large number of tasks), BB-ext and DA often require substantially less computation time than LA or LA-ext. Thus, the latter tests should be used only if BB-ext and DA fail or cannot be applied. As tardiness constraints become more relaxed (inset (c), Rule R3), the performance of LA and LA-ext increases dramatically. When a wide range of tardiness thresholds must be supported (inset (a), Rule R1), the performance of LA is sometimes worse than that of LA-ext. The reason for this is that more stringent (unmodified) task deadlines are assumed in the case of LA.

For NP-GEDF the obtained results are similar. As insets (d)–(f) show, NP-LA and NP-LA-ext are superior to NP and NP-DA. Again, due to their higher computational complexity, NP-LA and NP-LA-ext should only be used when NP or NP-DA fail or cannot be applied. NP-LA-ext exhibited the best performance under NP-GEDF. It outperformed NP-LA because non-preemptivity causes many of the "earlier" deadlines assumed in NP-LA to essentially be ignored.

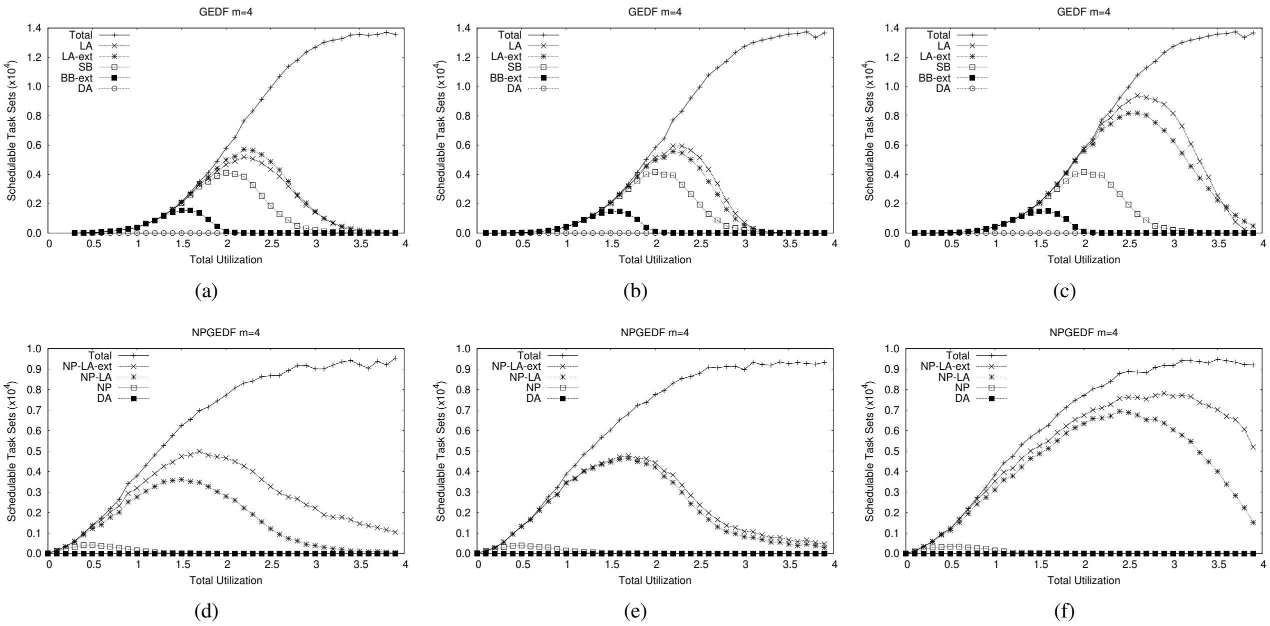

Figure 3. Comparison of schedulability of soft real-time task sets with implicit deadlines for tardiness thresholds under rules (a,d) R1, (b,e) R2, and (c,f) R3.

Is extending deadlines a good idea? Though showing good performance, LA-ext and NP-LA-ext require task deadlines to be extended. In an actual schedule, this may cause jobs to miss their original deadlines more frequently. Job deadlines define job *priorities* while tardiness thresholds define *timing requirements*. The manner in which these concepts are interpreted may be application-dependent. Indeed, in an actual application, tradeoffs may exist regarding how deadlines and tardiness thresholds are defined. Theorem 1 allows such tradeoffs to be explored, while the tests in [4, 5] do not.

5. Conclusion

We have presented a sufficient test for checking that arbitrary pre-defined deadline tardiness bounds are not violated under GEDF and NP-GEDF. These tests are of pseudo-polynomial time complexity if total utilization is less than the number of processors.

The experiments we conducted show that for randomly-generated task sets, our tests outperform known schedulability tests ([3, 4, 5, 8, 9]) due to their flexibility when considering individual task timing constraints. Unlike the test in [3], our NP-GEDF test does not restrict task parameters.

Several interesting avenues for further work exist. First, it would be interesting to obtain similar analysis for static-priority scheduling. Second, we would like to extend our analysis to preemptive tasks that may have non-preemptive code segments or self-suspend. Third, we would like to consider more sophisticated task models (e.g., as in [7]) and systems in which processors may be only partially available for executing real-time tasks.

Acknowledgment: We are grateful to Sanjoy Baruah for his valuable comments on earlier drafts of this paper.

References

[1] T. Baker. Multiprocessor EDF and deadline monotonic schedulability analysis. In *Proceedings of the 24th IEEE Real-Time Systems Symposium*, pages 120–129, December 2003.

[2] T. Baker. A comparison of global and partitioned EDF schedulability tests for multiprocessors. Technical Report TR-051101, Department of Computer Science, Florida State University, 2005.

[3] S. Baruah. The non-preemptive scheduling of periodic tasks upon multiprocessors. *Real-time Systems*, 32:9–20, 2006.

[4] S. Baruah. Techniques for multiprocessor global schedulability analysis. In *Proceedings of the IEEE Real-Time Systems Symposium*, December 2007.

[5] S. Baruah and T. Baker. Global EDF schedulability analysis of arbitrary sporadic task systems. In *Proceedings of the EuroMicro Conference on Real-Time Systems*, July 2008. To appear.

[6] M. Bertogna, M. Cirinei, and G. Lipari. Improved schedulability analysis of EDF on multiprocessor platforms. In *Proceedings of the 17th Euromicro Conference on Real-Time Systems*, pages 209–218, July 2005.

[7] S. Chakraborty and L. Thiele. A new task model for streaming applications and its schedulability analysis. In *Proceedings of the IEEE Design Automation and Test in Europe (DATE)*, pages 486–491, March 2005.

[8] U. Devi. *Soft Real-Time Scheduling on Multiprocessors*. PhD thesis, University of North Carolina, Chapel Hill, NC, 2006.

[9] U. Devi and J. Anderson. Tardiness bounds for global EDF scheduling on a multiprocessor. In *Proceedings of the 26th IEEE Real-Time Systems Symposium*, pages 330–341, December 2005.

[10] U. Devi and J. Anderson. Flexible tardiness bounds for sporadic real-time task systems on multiprocessors. In *Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium*, April 2006 (on CD ROM).

Appendix

In this appendix, we prove Lemma 7 and Theorem 2. Note that, in the proofs that follow, NP-GEDF is assumed to be the scheduler.

Lemma 7. *$I_{CH}(T_i)$ equals*

$$\begin{cases} \min(DBF'(T_i,\delta_k), \delta_k+\Theta_k-e_k+1), & i\neq k,\\ \min(DBF'(T_i,\delta_k)-e_k, \\ \qquad \max(\delta_k-D_k, \delta_k-p_k+\Theta_k)), & i=k, \end{cases}$$

where

$$DBF'(T_i,\delta_k) = \left\lfloor \frac{\delta_k+\Theta_i}{p_i} \right\rfloor \cdot e_i + \min(e_i, (\delta_k+\Theta_i)\ mod\ p_i)).$$

Proof. Consider $T_i \in \tau_{CH}$. There are two cases.
Case 1: $i \neq k$**.** The competing demand due to T_i will be maximized if we move the deadlines of all competing jobs so that the deadline of the last competing job coincides with t_d as shown in Fig. 4. Under these conditions, we let

$$d_{i,h} = t_d - q\cdot p_i \tag{12}$$

be the deadline of the earliest job so that $d_{i,h}+\Theta_i \geq t_0$. The job $T_{i,h}$ is thus the earliest job of T_i (potentially tardy), that may execute during $[t_0, t_d)$. The competing demand due to T_i's jobs executing within $[t_0, t_e) \cup \Gamma$ is thus bounded by the demand due to q jobs that have deadlines at or before t_d and are released at or after $r_{i,h}+p_i$, plus the demand imposed by the job $T_{i,h}$, which cannot exceed the smaller of e_i and the length of the interval $[t_0, d_{i,h}+\Theta_i)$, which is $t_d - q\cdot p_i + \Theta_i - t_0$. Formally,

$$\begin{aligned} I_{CH}(T_i) &= q\cdot e_i + \min(e_i, t_d - q\cdot p_i + \Theta_i - t_0)\\ &= q\cdot e_i + \min(e_i, \delta_k + \Theta_i - q\cdot p_i). \end{aligned} \tag{13}$$

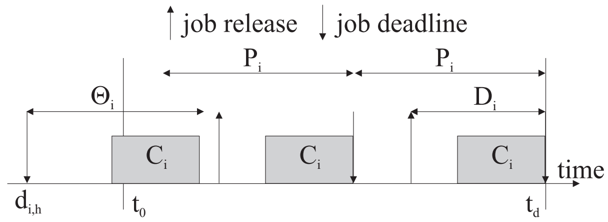

Figure 4. Computing $I_{CH}(T_i)$ when $T_i \in \tau_{CH}$.

To find q, observe that $d_{i,h}+\Theta_i \geq t_0$, which by (12) yields

$$\begin{aligned} & t_d - q\cdot p_i + \Theta_i \geq t_0\\ \Rightarrow\ & t_d - t_0 + \Theta_i \geq q\cdot p_i\\ \Rightarrow\ & \frac{\delta_k+\Theta_i}{p_i} \geq q\\ \Rightarrow\ & q = \left\lfloor \frac{\delta_k+\Theta_i}{p_i} \right\rfloor. \end{aligned}$$

Setting this expression for q into (13), we get

$$\begin{aligned} & I_{CH}(T_i)\\ = & \left\lfloor \frac{\delta_k+\Theta_i}{p_i} \right\rfloor \cdot e_i + \min(e_i, \delta_k+\Theta_i - q\cdot p_i)\\ = & \left\lfloor \frac{\delta_k+\Theta_i}{p_i} \right\rfloor \cdot e_i + \min(e_i, (\delta_k+\Theta_i)\ mod\ p_i). \end{aligned}$$

By Lemma 4, the required result follows.

Case 2: $i = k$**.** Repeating the reasoning from the previous case, we find that the total demand of jobs of T_k with deadlines at most t_d is at most $DBF'(T_k,\delta_k)$. However, the execution time of $T_{k,j}$ should be excluded, so we subtract e_k from $DBF'(T_k,\delta_k)$. By Lemma 4, the required result follows. □

The next three lemmas are used to prove Theorem 2.

Lemma 9. *For $\delta' \geq 0$, $DBF(T_i,\delta') \leq u_i\cdot\delta' + \max(0, u_i\cdot(p_i-D_i))$, and $DBF'(T_i,\delta') \leq u_i\cdot(\delta'+\Theta_i-e_i)+e_i$.*

Proof. By (7), $DBF(T_i,\delta') \leq \max(0, \left(\frac{\delta'-D_i}{p_i}+1\right)\cdot e_i) = \max(0, u_i\cdot(\delta'-D_i)+u_i\cdot p_i) \leq u_i\cdot\delta' + \max(0, u_i\cdot(p_i-D_i))$. To prove $DBF'(T_i,\delta') \leq u_i\cdot(\delta'+\Theta_i-e_i)+e_i$, we consider two cases.
Case 1: $e_i < (\delta'+\Theta_i)\ mod\ p_i$**.** In this case, $\left\lfloor \frac{\delta'+\Theta_i}{p_i} \right\rfloor < \frac{\delta'+\Theta_i-e_i}{p_i}$. Applying this inequality and the condition of the case in (8), we get $DBF'(T_i,\delta') = \left\lfloor \frac{\delta'+\Theta_i}{p_i} \right\rfloor \cdot e_i + e_i < \frac{\delta'+\Theta_i-e_i}{p_i}\cdot e_i + e_i = u_i\cdot(\delta'+\Theta_i-e_i)+e_i$.

Case 2: $e_i \geq (\delta' + \Theta_i) \; mod \; p_i$**.** In this case,

$$\left\lfloor \frac{\delta' + \Theta_i}{p_i} \right\rfloor \geq \frac{\delta' + \Theta_i - e_i}{p_i}. \tag{14}$$

As $(\delta' + \Theta_i) \; mod \; p_i = \delta' + \Theta_i - q \cdot p_i$, where $q = \left\lfloor \frac{\delta' + \Theta_i}{p_i} \right\rfloor$, by the condition of the case, we can rewrite (8) as,

$$\begin{aligned}
&DBF'(T_i, \delta') \\
&\quad = \quad q \cdot e_i + \delta' + \Theta_i - q \cdot p_i \\
&\quad = \quad \delta' + \Theta_i - (p_i - e_i) \left\lfloor \frac{\delta' + \Theta_i}{p_i} \right\rfloor \\
&\quad \qquad \{\text{by (14)}\} \\
&\quad \leq \quad \delta' + \Theta_i - (p_i - e_i) \cdot \frac{\delta' + \Theta_i - e_i}{p_i} \\
&\quad = \quad e_i + (\delta' + \Theta_i - e_i) \cdot \left(1 - \frac{p_i - e_i}{p_i}\right) \\
&\quad = \quad u_i \cdot (\delta' + \Theta_i - e_i) + e_i.
\end{aligned}$$

□

Lemma 10. *$I_{CL}(T_i) \leq e_i$. For $\delta' \geq 0$, $I_{NC}(T_i) \leq DBF(T_i, \delta') \leq u_i \cdot \delta' + \max(0, u_i \cdot (p_i - D_i))$ and $I_{CH}(T_i) \leq DBF'(T_i, \delta') \leq u_i \cdot (\delta' + \Theta_i - e_i) + e_i$. If $\Theta_k = x + e_k$, then $I_{CH}(T_k) \leq DBF'(T_k, \delta') - e_k \leq u_k \cdot (\delta' + \Theta_k - e_k)$.*

Proof. Follows directly follow from Lemmas 5, 6, 7, and Lemma 9. □

Lemma 11. *$\sum_{T_i \in \tau_{CH} \cup \tau_{CL}} e_i \leq E(m)$, $\sum_{T_i \in \tau_{CH}} u_i \leq U(m-1)$, and $\sum_{T_i \in \tau_{NC} \cup \tau_{CH}} u_i \leq U_{sum}$.*

Proof. Follows from Def. 7 and the constraints in (10). □

Theorem 2. *If (11) holds for task T_k for all $\delta_k \in [\max(\min(D_i), \min(D_k, p_k - \Theta_k)), \delta_{max}]$, then (11) holds for all $\delta_k \geq \max(\min(D_i), \min(D_k, p_k - \Theta_k))$.*

Proof. Suppose that (11) does not hold for some task T_k and some $\delta' > \delta_{max}$. Without loss of generality, we can assume $\delta' \geq D_k$. Then, there exist $\tau_{CH}, \tau_{CL}, \tau_{NC}$ subject to (10) such that

$$\begin{aligned}
\sum_{T_i \in \tau_{CH}} I_{CH}(T_i) \quad &+ \quad \sum_{T_i \in \tau_{CL}} I_{CL}(T_i) + \sum_{T_i \in \tau_{NC}} I_{NC}(T_i) \\
&\geq \quad m \cdot (\delta' + \Theta_k - e_k + 1).
\end{aligned}$$

Applying Lemma 10 to this inequality, we get

$$\begin{aligned}
&\sum_{T_i \in \tau_{CH}} (u_i \cdot (\delta' + \Theta_i - e_i) + e_i) + \sum_{T_i \in \tau_{CL}} e_i \\
&\quad + \sum_{T_i \in \tau_{NC}} (u_i \cdot \delta' + \max(0, u_i \cdot (p_i - D_i))) \\
&\qquad\qquad \geq m \cdot (\delta' + \Theta_k - e_k + 1).
\end{aligned}$$

After re-grouping, we get

$$\begin{aligned}
\delta' \cdot \sum_{T_i \in \tau_{CH} \cup \tau_{NC}} u_i + \sum_{T_i \in \tau_{CH}} u_i \cdot \Theta_i + \sum_{T_i \in \tau_{CH} \cup \tau_{CL}} e_i& \\
+ \sum_{T_i \in \tau_{NC}} \max(0, u_i \cdot (p_i - D_i))& \\
\geq m \cdot (\delta' + \Theta_k - e_k + 1).&
\end{aligned}$$

Applying Lemma 11 to this inequality, we get

$$\begin{aligned}
U_{sum} \cdot \delta' \quad &+ \quad U(m-1) \cdot \max(\Theta_i) + E(m) + R \\
&\geq \quad m \cdot (\delta' + \Theta_k - e_k + 1),
\end{aligned}$$

where R is defined as in Def. 7. Solving the inequality above for δ', we get $\delta' \leq (E(m) + U(m-1) \cdot \max(\Theta_i) + R + m \cdot (e_k - \Theta_k - 1))/(m - U_{sum})$, which contradicts the assumption that $\delta' > \delta_{max}$ holds. □