Push Forward: Global Fixed-Priority Scheduling of Arbitrary-Deadline Sporadic Task Systems

Jian-Jia Chen
TU Dortmund University, Germany
jian-jian.chen@tu-dortmund.de
0000-0001-8114-9760

Georg von der Brüggen
TU Dortmund University, Germany
georg.von-der-brueggen@tu-dortmund.de
0000-0002-8137-3612

Niklas Ueter
TU Dortmund University, Germany
niklas.ueter@tu-dortmund.de
0000-0002-6722-4805

Abstract

The sporadic task model is often used to analyze recurrent execution of tasks in real-time systems. A sporadic task defines an infinite sequence of task instances, also called jobs, that arrive under the minimum inter-arrival time constraint. To ensure the system safety, timeliness has to be guaranteed in addition to functional correctness, i.e., all jobs of all tasks have to be finished before the job deadlines. We focus on analyzing arbitrary-deadline task sets on a homogeneous (identical) multiprocessor system under any given global fixed-priority scheduling approach and provide a series of schedulability tests with different tradeoffs between their time complexity and their accuracy. Under the arbitrary-deadline setting, the relative deadline of a task can be longer than the minimum inter-arrival time of the jobs of the task. We show that global deadline-monotonic (DM) scheduling has a speedup bound of $3 - 1/M$ against any optimal scheduling algorithms, where M is the number of identical processors, and prove that this bound is asymptotically tight.

2012 ACM Subject Classification C.3[**Computer Systems Organization**]: *Special-Purpose and Application-Based Systems* - Real-Time and Embedded Systems;
I.1.2[**Computing Methodologies**]: *Algorithms* - Analysis of Algorithms

Keywords and phrases global fixed-priority scheduling, schedulability analyses, speedup bounds

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2018.8

Related Version https://ls12-www.cs.tu-dortmund.de/daes/media/documents/publications/downloads/2018-chen-ecrts-push-forward-full.pdf

Funding This paper is supported by DFG, as part of the Collaborative Research Center SFB876 (http://sfb876.tu-dortmund.de/), project B2.

1 Introduction

The sporadic task model is the basic task model in real-time systems, where each task τ_i releases an infinite number of *task instances* (*jobs*) under its *minimum inter-arrival time* (*period*) T_i and is further characterized by its *relative deadline* D_i and its *worst-case execution time* C_i. The sporadic task model has been widely adopted in real-time systems.

A sporadic task defines an infinite sequence of task instances, also called *jobs*, that arrive under the minimum inter-arrival time constraint, i.e., any two consecutive releases of jobs of task τ_i are temporally separated by at least T_i. When a job of task τ_i arrives at time t, it must finish no later than its *absolute deadline* $t + D_i$. If all tasks release their jobs strictly periodically with period T_i, the task model is the well-known Liu and Layland task model [33]. A sporadic task set is called with 1) *implicit deadlines*, if the relative deadlines are equal to their minimum inter-arrival times, 2) *constrained deadlines*, if the minimum inter-arrival times are no less than their relative deadlines, and 3) *arbitrary deadlines*, otherwise.

To schedule such task sets on a multiprocessor platform, three paradigms have been widely adopted: partitioned, global, and semi-partitioned multiprocessor scheduling. The *partitioned* scheduling approach partitions the tasks statically among the available processors, i.e., a task executes all its jobs on the assigned processor. The *global* scheduling approach allows a job to migrate from one processor to another at any time. The *semi-partitioned* scheduling approach decides whether a task is divided into subtasks statically and how each task/subtask is then assigned to a processor. A comprehensive survey of multiprocessor scheduling for real-time systems can be found in [23].

We focus on *global fixed-priority preemptive scheduling* on M identical processors, i.e., unique fixed priority levels are statically assigned to the tasks and at any point in time the M highest-priority jobs in the ready queue are executed. Hence, the schedule is *workload-conserving*. The response time of a job is defined as its finish time minus its arrival time. The worst-case response time of a task is an upper bound on the response times of all the jobs of the task and can be derived by a *(worst-case) response time analysis* for a sporadic task under a given scheduling algorithm. Verifying whether a set of sporadic tasks can meet their deadlines by a scheduling algorithm is called a *schedulability test*, i.e., verifying if the *(worst-case) response time* is smaller than or equal to the *relative deadline*.

1.1 Related Work

For uniprocessor systems, i.e, M=1, the exact schedulability test and the (tight) worst-case response time analysis by using *busy intervals* were provided by Lehoczky [32]. Several approaches have been proposed to reduce the time complexity, e.g., [35]. Bini and Buttazzo [12] proposed a framework of schedulability tests that can be tuned to balance the time complexity and the acceptance ratio of the schedulability test for uniprocessor sporadic task systems. To achieve polynomial-time schedulability tests and response time analyses, Lehoczky [32] proposed a utilization upper bound for a set of sporadic arbitrary-deadline tasks under fixed-priority scheduling. The linear-time response-time bound for fixed-priority systems was first proposed by Davis and Burns [22], and later improved by Bini et al. [14, 15] and Chen et al. [18]. The computational complexity of the schedulability test problem and the worst-case response time analysis in uniprocessor systems for different variances can be found in [16, 25, 24, 27, 26].

In this paper, we will implicitly assume multiprocessor systems, i.e., $M \geq 2$. Many results are known for constrained-deadline ($D_i \leq T_i$) and implicit-deadline task systems ($D_i = T_i$) on identical multiprocessor platforms, e.g., [2, 5, 30, 1, 7, 18]. For details, please refer to the survey by Davis and Burns [23]. Unfortunately, deriving exact schedulability tests under multiprocessor global scheduling is much harder than deriving them for uniprocessor systems due to the lack of concrete worst-case scenarios that can be constructed efficiently. Most results in the literature focus on sufficient schedulability tests. Exceptions are the exhaustive search under discrete time parameters by Baker and Cirinei [4], finite automata under discrete time parameters by Geeraerts et al. [29], and hybrid finite automata by

Sun and Lipari [36]. Specifically, Geeraerts et al. [29] showed that the schedulability test formulation by Baker and Cirinei [4] is PSPACE-Complete.

Regarding global fixed-priority scheduling for arbitrary-deadline task systems, several sufficient schedulability tests and safe worst-case response time analyses have been proposed, e.g., [3, 4, 8, 9, 30, 37, 31]. Baker [3] designed a test based on certain properties to characterize a *problem window*. Baruah and Fisher [8, 9] used different annotations to extend the analysis window and derived corresponding exponential-time schedulability tests. The first worst-case response-time analysis for arbitrary-deadline task systems was proposed by Guan et al. [30], where the authors used the insight proposed by Baruah [5] to limit the number of carry-in jobs, and then apply the workload function proposed by Bertogna et al. [11] to quantify the requested demand of higher-priority tasks. Unfortunately, it has recently been shown by Sun et al. [37] that this analysis in [30] is optimistic. In addition, Sun et al. [37] derived a complex carry-in workload function for the response time analysis where all possible combinations of carry-in and non-carry-in functions have to be explicitly enumerated. However, their method is computationally intractable since the time complexity is exponential. Huang and Chen [31] proposed a more precise quantification for the number of carry-in jobs of a task than the bounds used in the tests provided in [3, 9]. They also presented a response time bound for arbitrary-deadline tasks under global scheduling in multiprocessor systems with linear-time complexity.

1.2 Our Contribution

We consider arbitrary-deadline sporadic task systems, which is the most general case of the sporadic real-time task model. To quantify the performance loss due to efficient schedulability tests and the non-optimality of scheduling algorithms, we will adopt the notion of speedup factors/bounds, also known as resource augmentation factors/bounds. Table 1 summarizes the state-of-the-art speedup bounds for the global deadline-monotonic (DM) scheduling, one specific global fixed-priority scheduling algorithm. Under global DM, a task τ_i has higher priority than task τ_j if $D_i \leq D_j$, in which ties are broken arbitrarily. The authors note that the proof by Lundberg [34] seems incomplete. However, the concrete task set in [34] provides the lower bound 2.668 of the speedup factors for global DM. Moreover, Andersson [1] showed that global slack monotonic scheduling has a speedup bound of $\frac{3+\sqrt{5}}{2} \approx 2.6181$ for implicit-deadline task systems. However, no better global fixed-priority scheduling algorithms with respect to speedup factors are known for constrained-deadline and arbitrary-deadline task systems.

Our Contributions: Table 1 summarizes the related results and the contribution of this paper for multiprocessor global fixed-priority preemptive scheduling. We improve the best known results by Baruah and Fisher [8] with respect to the speedup bounds. Our contributions are:

- For *any* global fixed-priority preemptive scheduling, we provide a series of schedulability tests with different tradeoffs between time complexity and accuracy in Section 3 and Section 4.
- We show that the global deadline-monotonic scheduling algorithm has a speedup factor $3-1/M$ with respect to the optimal multiprocessor scheduling policies when considering task systems with arbitrary deadlines. This improves the analyses by Fisher and Baruah with respect to the speedup bounds, i.e., $4-1/M$ [9] and 3.73 [8].
- We show that all the schedulability tests we provide in this paper analytically dominate the tests by Baruah and Fisher [8] for global DM. We also show that global DM has a

		implicit deadlines	constrained deadlines	arbitrary deadlines
Global DM	upper bounds	2.668 [34] (poly.-time)	$3-1/M$ [7] (expo.-time)	$\frac{2(M-1)}{4M-1-\sqrt{12M^2-8M+1}} \leq 3.73$ [8] (expo.-time)
		2.823 [18] (poly.-time)	$3-1/M$ [18] (poly.-time)	$3-\frac{1}{M}$ (*this paper*) (poly.-time)
	lower bounds	2.668 [34]	2.668 [34]	2.668 [34]
				$3-\frac{3}{M+1}$ (this paper)

Table 1 Speedup bounds of the global deadline-monotonic (DM) scheduling algorithm for sporadic task systems.

speedup lower bound of $3-3/(M+1)$, which shows that our schedulability analyses are asymptotically tight with respect to the speedup factors.

2 System Model, Definitions, and Assumptions

We consider an arbitrary-deadline sporadic task set $\mathbf{T}$ with N tasks executed on $M \geq 2$ identical processors based on global fixed-priority preemptive scheduling. We assume that the priority levels of the tasks are unique (and given) and that τ_i has higher priority than task τ_j if $i < j$. When there is only one processor, i.e., $M = 1$, the existing results discussed in Section 1.1 can be adopted, and our analysis here cannot be applied. We will implicitly use the assumption $M \geq 2$ in the paper.

By definition, M is an integer. In addition to C_i, T_i, D_i, we also define the utilization U_i task τ_i as C_i/T_i. We will implicitly assume that $D_i > 0$, $C_i > 0$, $T_i > 0$, $C_i/D_i \leq 1$, and $U_i \leq 1 \; \forall \tau_i$ in this paper. Moreover, *intra-task parallelism* is not allowed. *At most one job* of task τ_i can be executed on at most one processor at each instant in time, *regardless of the number of the jobs of task τ_i awaiting for execution and the number of idle processors*. We denote the set of natural numbers as $\mathbb{N}$.

2.1 Resource Augmentation

We assume the original platform speed is 1. Therefore, running the platform at speed s implies that the worst-case execution time of task τ_i becomes C_i/s. A scheduling algorithm $\mathcal{A}$ has a *speedup bound* s with respect to the optimal schedule, if it guarantees to always produce a feasible solution when 1) each processor is sped up to run at s times of the original speed of the platform and 2) the task set $\mathbf{T}$ can be feasibly scheduled on the original M identical processors, i.e., running at speed 1.

We will use the negation of the above definition to quantify the failure of algorithm $\mathcal{A}$: *If $\mathcal{A}$ fails to ensure that all the tasks in $\mathbf{T}$ meet their deadlines, then no feasible multiprocessor schedule exists when each processor is slowed down to run at speed $1/s$.*

2.2 Definitions and Necessary Condition

We define the following notation according to the task system and the priority assignment:

- density δ_i of task τ_i: $\delta_i = C_i/\min\{D_i, T_i\}$
- maximum density $\delta_{\max}(k)$ among the first k tasks: $\delta_{\max}(k) = \max_{i=1}^{k} \delta_i$
- maximum between the utilization of the higher-priority tasks and the density of task τ_k: $U_{\delta,k}^{\max} = \max\{\max_{i=1}^{k-1} U_i, \delta_k\}$
- demand bound function [10] $\text{DBF}(\tau_i, t)$ of task τ_i, further explained in Definition. 2.1
- load $\text{LOAD}(k)$ of the first k tasks: $\text{LOAD}(k) = \max_{t>0} \frac{\sum_{i=1}^{k} \text{DBF}(\tau_i,t)}{t}$

▶ **Definition 2.1** (demand bound function (DBF) by Baruah [10]). For any $t \geq 0$

$$\mathrm{DBF}(\tau_i, t) = \max\left\{0, \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right) C_i\right\} \tag{1}$$

The demand bound function $\mathrm{DBF}(\tau_i, t)$ defines the execution time task τ_i must finish for any interval length t to ensure its timing correctness. □

Since $\delta_i \geq U_i$ by definition, we know that $U^{\max}_{\delta,k} \leq \delta_{\max}(k)$. As we assume $C_i/D_i \leq 1$ and $U_i \leq 1$ we know that $\delta_i \leq 1$. In addition to DBFs, we will heavily use the following workload function:

▶ **Definition 2.2** (Workload function). Let $work_i(t)$ be a workload function, representing the maximum amount of time for *sequentially* executing the jobs of task τ_i released in time interval $[a, a+t)$, i.e., jobs released before a are not considered. For any $t \geq 0$

$$work_i(t) = \left\lfloor \frac{t}{T_i} \right\rfloor C_i + \min\left\{C_i, t - \left\lfloor \frac{t}{T_i} \right\rfloor T_i\right\}. \tag{2}$$

For notational brevity, we set $work_i(t)$ to $-\infty$ if $t < 0$. □

The workload function $work_i(t)$ defined above is a piecewise function, i.e., linear in intervals $[\ell T_i, \ell T_i + C_i]$ with a slope 1 and constant, $(\ell+1)C_i$, in intervals $[\ell T_i + C_i, (\ell+1)T_i]$ for any non-negative integer ℓ. Two examples of the workload function are illustrated in Figure 2 in Section 3. To prove the speedup bound, we will utilize the following necessary condition.

▶ **Lemma 2.3.** *A task set* $\mathbf{T}$ *with* N *tasks is not schedulable by any multiprocessor scheduling algorithm when the* M *processors are running at any speed* s*, if*

$$\max\left\{\max_{t>0} \frac{\sum_{\tau_i \in \mathbf{T}} \mathrm{DBF}(\tau_i, t)}{Mt}, \frac{\sum_{\tau_i \in \mathbf{T}} U_i}{M}, \delta_{\max}(N)\right\} > s. \tag{3}$$

Proof. This is widely used based on a reformulation in the literature, e.g., [8, 9]. ◀

2.3 Analysis Based on DBFs

Baruah and Fisher in [8] provided a schedulability test for task τ_k under global deadline-monotonic (DM) scheduling that is based on the Demand Bound Functions (DBF), assuming that the tasks are sorted according to DM order already, i.e., $D_1 \leq D_2 \leq \ldots \leq D_N$:

▶ **Theorem 2.4** (Baruah and Fisher [8], revised in [17]). *Let* μ_k *be defined as* $M - (M-1)\delta_{\max}(k)$. *Task* τ_k *is schedulable under global DM if* [1]

$$2\mathrm{LOAD}(k) + (\lceil \mu_k \rceil - 1)\delta_{\max}(k) \leq \mu_k. \tag{4}$$

3 Schedulability Test by Pushing Forward

In this section, we provide several conditions for the schedulability of task τ_k under a given preemptive global fixed-priority scheduling algorithm. They lead to a sufficient schedulability test for τ_k, assuming that the schedulability of the tasks $\tau_1, \tau_2, \ldots, \tau_{k-1}$ under the given

[1] The original proof by Baruah and Fisher [8] had a mathematical flaw in their Lemma 3, i.e., setting μ_k to $M - (M-1)\delta_k$. It can be fixed by setting μ_k to $M - (M-1)\delta_{\max}(k)$.

algorithm is already verified. This means that for all tasks τ_i with $i < k$ the worst-case response time is at most D_i. Therefore, the test should be applied for all tasks, i.e., from the highest-priority task to the lowest-priority task, to ensure the schedulability of the task set under the (specified/given) global fixed-priority scheduling. As the test presented here has a high time complexity, we provide more efficient tests in Section 4.

3.1 Analysis Window Extension

We analyze the schedulability of τ_k by looking at the intervals where τ_k is active in the schedule S provided by the global fixed-priority scheduling algorithm according to the following definition:

▶ **Definition 3.1 (active task).** For a schedule S, a task τ_i is active at time t, if there is (at least) one job of τ_i that has arrived before or at t and has not finished yet at time t. ◻

The schedulability conditions are proved by using *contrapositive*. Suppose a schedule S produced by the given global fixed-priority scheduling algorithm and that t_d is the earliest (absolute) deadline at which a job of task τ_k misses its deadline. Let t_a be the time instant in S such that τ_k is continuously active in the time interval $[t_a, t_d)$ and is not active *immediately* prior to t_a. By definition, t_a must be the arrival time of a job of task τ_k. Suppose that t_d is the absolute deadline of the ℓ-th job of task τ_k that arrived in the time interval $[t_a, t_d)$. Therefore, as τ_k is a sporadic task, $t_d - t_a \geq (\ell - 1)T_k + D_k$. For notational brevity, we define $D'_k = (\ell - 1)T_k + D_k$ and $C'_k = \ell C_k$.

We remove all the jobs of task τ_k that arrive before t_a and all the jobs with priorities lower than τ_k from the schedule S. The schedule of task τ_k remains unchanged in the resulting (new) schedule S, due to the preemptiveness of the global fixed-priority scheduling algorithm. Let C^*_k be the amount of time that task τ_k is executed from t_a to t_d. Since the ℓ-th job of task τ_k misses its deadline, we know that $C^*_k < \ell C_k = C'_k$. We now introduce three functions that are defined for any $t \leq t_d$.

- Let $E(t, t_d)$ be the amount of workload (sum of the execution times) of the higher-priority jobs, i.e., from $\tau_1, \tau_2, \ldots, \tau_{k-1}$, *executed* in the time interval $[t, t_d)$ in schedule S.
- Let $W(t, t_d)$ be $C^*_k + E(t, t_d)$.
- Let $\Omega(t, t_d)$ be $\frac{W(t,t_d)}{t_d - t}$.

Those definitions and the deadline miss of task τ_k at time t_d lead to the following lemma.

▶ **Lemma 3.2.** *Since τ_k misses its deadline at t_d in S, the following conditions hold:*

$$E(t_a, t_d) \geq M \times (t_d - t_a - C^*_k) \tag{5}$$

$$W(t_a, t_d) > M \times (t_d - t_a) - (M - 1)C'_k \tag{6}$$

$$\Omega(t_a, t_d) > M - (M - 1) \times \frac{C'_k}{D'_k} \tag{7}$$

Proof. Since task τ_k is active from t_a to t_d and is only executed for *exactly* C^*_k amount of time, we know that all M processors must be busy executing other higher-priority jobs for at least $t_d - t_a - C^*_k$ amount of time. Therefore, the amount of workload $E(t_a, t_d)$ of the higher-priority jobs executed in the time interval $[t_a, t_d)$ must be at least $M \times (t_d - t_a - C^*_k)$,

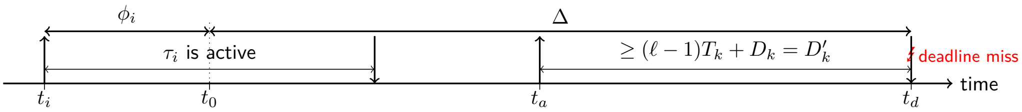


Figure 1 The notation used in Section 3: 1) task τ_k is continuously active from t_a to t_d with a deadline miss at time t_d; 2) time instant t_0 is the smallest value of $t \leq t_a$ such that $\Omega(t, t_d) \geq \mu_k$; 3) time instant t_i is the arrival time of a higher-priority carry-in task τ_i if τ_i is continuously active in time interval $[t_i, t_0 + \varepsilon]$, where $t_i < t_0$ and $\varepsilon > 0$ is an arbitrarily small number; 4) ϕ_i is $t_0 - t_i$ and Δ is $t_d - t_0$.

i.e., Eq. (5) must hold.[2] Therefore, since $W(t_a, t_d)$ is defined as $E(t_a, t_d) + C_k^*$, we have

$$W(t_a, t_d) \geq M \times (t_d - t_a - C_k^*) + C_k^* > M \times (t_d - t_a) - (M-1)C'_k,$$

where the last inequality is due to $M \geq 2$ and $C'_k > C_k^*$. This leads to the conditions in Eq. (6). Since $\Omega(t_a, t_d)$ is defined as $\frac{W(t_a,t_d)}{t_d - t_a}$ and $D'_k \leq t_d - t_a$, we have

$$\Omega(t_a, t_d) \geq M - (M-1)\frac{C'_k}{t_d - t_a} \geq M - (M-1)\frac{C'_k}{D'_k},$$

i.e., the condition in Eq. (7). ◀

Although the interval $[t_a, t_d)$ can already be used for constructing the schedulability tests, researchers have tried to push the interval of interest towards $[t_0, t_d)$ for some $t_0 \leq t_a$ based on certain properties, e.g., [31, 9, 8]. Such extensions have been shown to provide better quantifications of the interfering workload from the higher-priority tasks. In our analysis, we will use a similar extension strategy as suggested by Baruah and Fisher [8] based on a user-specified parameter ρ.

The following definition and lemmas are from [8]. Figure 1 provides an illustration of our notation based on the above definitions.

▶ **Definition 3.3.** Suppose that $\mu_k = M - (M-1)\rho$ for a certain ρ with $1 \geq \rho \geq \frac{C'_k}{D'_k}$. For the schedule S, let time instant t_0 be the smallest value of $t \leq t_a$ such that $\Omega(t, t_d) \geq \mu_k$. This means, $\Omega(t, t_d) < \mu_k$ for any $t < t_0$. □

▶ **Lemma 3.4.** *If τ_k misses its deadline at t_d, for any ρ with $1 \geq \rho \geq \frac{C'_k}{D'_k}$, the time t_0, as defined in Definition 3.3, always exists with $\Omega(t_0, t_d) \geq \mu_k$ and $t_0 \leq t_a$.*

Proof. By Eq. (7) from Lemma 3.2 and $\rho \geq \frac{C'_k}{D'_k}$, we know

$$\Omega(t_a, t_d) > M - (M-1) \times \frac{C'_k}{D'_k} \geq M - (M-1)\rho = \mu_k.$$

Therefore, such a time instant $t_0 \leq t_a$ exists, at least when the system starts. ◀

▶ **Definition 3.5 (carry-in task).** A task τ_i is a carry-in task in the schedule S, if τ_i is continuously active in a time interval $[t_i, t_0 + \varepsilon]$, for $t_i < t_0$ and an arbitrarily small $\varepsilon > 0$. □

▶ **Lemma 3.6.** *For $1 \geq \rho \geq \frac{C'_k}{D'_k}$, there are at most $\lceil M - (M-1)\rho \rceil - 1$ carry-in tasks at t_0 in schedule S.*

[2] The condition in Eq. (5) is widely used in the form of $E(t_a, t_d) > M \times (t_d - t_a - \ell C_k)$. Here, since we will use C_k^*, the correct form is with $\geq$.

3.2 Analysis Based on Workload Functions

By extending the interval of interest to $[t_0, t_d)$, Baruah and Fisher provided the schedulability test shown in Theorem 2.4 in this paper. However, they analyzed the workload in $[t_0, t_d)$ based on the DBFs by using the function LOAD(k) as an approximation, which will be shown pessimistic in Corollary 5.1 in Section 5. Moreover, their final analysis can only be applied for global DM. We will carefully analyze the workload executed in $[t_0, t_d)$ to ensure that the analytical accuracy is better preserved and that the analysis can be used for any global fixed-priority preemptive scheduling. We will demonstrate that our analysis dominates the analysis by Baruah and Fisher [8] in Corollary 5.1.

For the analysis before Theorem 3.10, we will assume that ρ is given and t_0 is already defined. According to Lemma 3.6, at time t_0 at most $\lceil M-(M-1)\rho\rceil-1$ tasks are active in schedule S. We quantify their contribution to the *executed* workload in time interval $[t_0, t_d)$ with two different forms from Lemma 3.7, denoted by $\omega_i^{heavy}(t_d - t_0)$, and from Lemma 3.8, denoted by $\omega_i^{light}(t_d - t_0)$. While Lemma 3.7 can be used in general, Lemma 3.8 only holds if $U_i \le \rho$. After these workload functions are detailed and explained, we will show their relationship in Lemma 3.9. Then, we will explain how they can be used and detail the constructed schedulability test in Theorem 3.10 based on the above concepts.

▶ **Lemma 3.7.** *If all jobs of a higher-priority task τ_i meet their deadlines, the upper bound $\omega_i^{heavy}(\Delta)$ on the workload of task τ_i executed from t_0 to t_d with $\Delta = t_d - t_0$ in schedule S is at most:*

$$\omega_i^{heavy}(\Delta) = work_i(\Delta + D_i). \tag{8}$$

Proof. Since all jobs of τ_i meet their deadlines, the jobs of τ_i executed in $[t_0, t_d)$ must arrive in the time interval $(t_0 - D_i, t_d)$. Therefore, the workload of task τ_i that can be sequentially executed is upper bounded by the workload function with length $t_d - (t_0 - D_i) = \Delta + D_i$. ◀

The key improvement achieved in this paper is due to the following Lemma 3.8 to safely bound the workload of a light task.

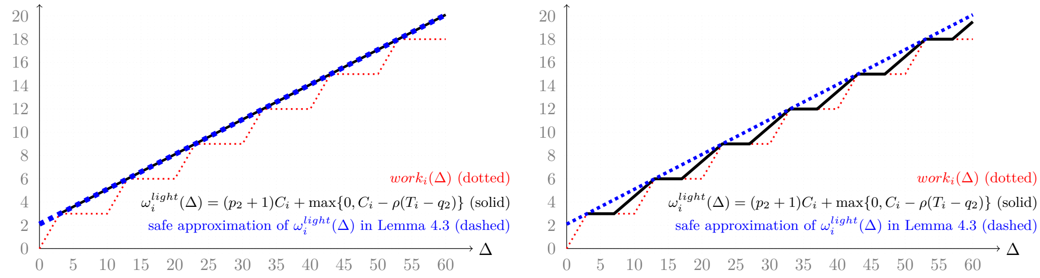
Figure 2 demonstrates the workload function for different cases in Lemma 3.8, together with a linear approximation that will be presented in Lemma 4.3. For the workload function defined in Eq. (9), informally speaking, the workload defined by $(p_2+1)C_i + \max\{0, C_i - \rho(T_i - q_2)\}$ can be imagined as if 1) there is an offset for C_i amount of execution time at beginning of the interval, and 2) the workload in each period starting from $C_i + p_2T_i$ to $C_i + (p_2+1)T_i$ is pushed to the end of the period with a slope ρ. For example, in Figure 2(b), the offset is 3, the workload increases from 3 at time 7 to 6 at time 13 with a slope $\rho = 0.5$, the workload increases from 6 at time 17 to 9 at time 23 with a slope $\rho = 0.5$, etc.

▶ **Lemma 3.8.** *If all jobs of a higher-priority task τ_i meet their deadlines and $U_i \le \rho \le 1$, the upper bound $\omega_i^{light}(\Delta)$ on the workload of task τ_i executed from t_0 to t_d with $\Delta = t_d - t_0$ in schedule S is:*

$$\omega_i^{light}(\Delta) = \begin{cases} \Delta & \text{if } 0 < \Delta \le C_i \\ \max\begin{cases} work_i(\Delta), \\ (p_2+1)C_i + \max\{0, C_i - \rho(T_i - q_2)\} \end{cases} & \text{if } \Delta > C_i \end{cases} \tag{9}$$

where $p_2 = \lceil(\Delta - C_i)/T_i\rceil - 1$ and q_2 is $\Delta - C_i - p_2T_i$.

Proof. As the case when $0 < \Delta \le C_i$ is due to the definition, let $\Delta > C_i$ for the rest of the proof. Based on the schedule S, let $t_i < t_0$ be the time instant such that task τ_i is



(a) $U_i = 0.3$ and $\rho = 0.3$ **(b)** $U_i = 0.3$ and $\rho = 0.5$

Figure 2 Two examples for the approximation of $work_i$ for τ_i with $T_i = 10, C_i = 3, D_i = 45$: black curves for $\omega_i^{light}(\Delta)$ defined in Lemma 3.8 and the approximation in Lemma 4.3 (blue curves).

continuously active in the time interval $[t_i, t_0]$ and task τ_i is not active *immediately* prior to t_i. If t_i does not exist, then task τ_i does not have workload released before t_0 that is still active. Therefore, the worst-case workload is $work_i(\Delta)$ in this case.

Let ϕ_i be $t_0 - t_i$. By the definition of t_i, if it exists, there are at most $\left\lceil \frac{\phi_i}{T_i} \right\rceil$ jobs of task τ_i executed in time interval $(t_i, t_0]$. For the rest of the proof, we only consider that t_i exists and that $\Delta > C_i$. By definition, t_i must be the arrival time of a job of task τ_i. Moreover, due to the definition of t_0 in Definition 3.3, we know that $\Omega(t_i, t_d) < M - (M-1)\rho$. Since $\Omega(t_i, t_d) < M - (M-1)\rho$ and $\Omega(t_0, t_d) \geq M - (M-1)\rho$, we have

$$W(t_0, t_d) = \Omega(t_0, t_d) \cdot (t_d - t_0) \geq (t_d - t_0)\mu_k = \Delta\mu_k \quad (10)$$

$$W(t_i, t_d) = \Omega(t_i, t_d) \cdot (t_d - t_i) < (t_d - t_i)\mu_k = (\Delta + \phi_i)\mu_k \quad (11)$$

Substracting Eq. (11) by Eq. (10), we have $W(t_i, t_d) - W(t_0, t_d) < \phi_i\mu_k$, i.e., in schedule S the workload executed in time interval $[t_i, t_0)$ is *strictly less* than $\phi_i\mu_k$. Suppose that y_i is the amount of time that task τ_i is executed in time interval $[t_i, t_0)$, i.e., task τ_i is active but blocked by other higher-priority jobs for $\phi_i - y_i$ amount of time in this time interval. When task τ_i is blocked in global fixed-priority scheduling, all the M processors are executing other jobs. The workload executed in time interval $[t_i, t_0)$ is at least $M(\phi_i - y_i) + y_i$. Therefore, by the above discussions, we know that

$$M(\phi_i - y_i) + y_i < \phi_i\mu_k = \phi_i(M - (M-1)\rho) \Rightarrow y_i > \rho\phi_i, \quad (12)$$

since $M \geq 2$. At time t_0, the remaining execution time of the jobs of task τ_i that arrived before t_0 in schedule S is at most $\lceil \phi_i/T_i \rceil C_i - \rho\phi_i$. Note that the existence of t_i in our definition means that $\lceil \phi_i/T_i \rceil C_i - y_i > 0$, i.e., $\lceil \phi_i/T_i \rceil C_i - \rho\phi_i > 0$.

The workload of task τ_i that is executed in the time interval $[t_i, t_d)$ in schedule S is at most $work_i(t_d - t_i) = work_i(\Delta + \phi_i)$. The workload of task τ_i that is executed in the time interval $[t_i, t_0)$ is at least $y > \rho\phi_i$. Therefore, the workload of task τ_i that is executed in the time interval $[t_0, t_d)$ in schedule S is upper bounded by $work_i(\Delta + \phi_i) - \rho\phi_i$.

The rest of the proof is to provide an upper bound of $work_i(\Delta+\phi_i) - \rho\phi_i$ for any arbitrary $\phi_i > 0$. The proof involves some detailed manipulations of the workload function. Before proceeding, we explain two basic properties of the workload function here by inspecting the periodicity of the workload function $work_i(t)$ where $p = \lfloor t/T_i \rfloor$, a non-negative integer:

- For $t = pT_i + x$ with $0 \leq x$, the recursion $work_i(pT_i + x) = pC_i + work_i(x)$ holds.
- For $t = pT_i + x$ with $0 \leq x \leq C_i$, the simplification $work_i(pT_i + x) = pC_i + x$ holds.

To identify the exact value of $work_i(\Delta + \phi_i)$, we define the following variables p_1, p_2, q_1, and q_2 for brevity:

- Let p_1 be $\lceil \phi_i/T_i \rceil - 1$ and q_1 be $\phi_i - p_1 T_i$, i.e., $p_1 + 1$ is the number of jobs of task τ_i that can be released in $[t_i, t_0]$. By definition $\phi_i > 0$, which implies that p_1 is a non-negative integer, $0 < q_1 \leq T_i$, and $\phi_i = p_1 T_i + q_1$.
- Let p_2 be $\lceil (\Delta - C_i)/T_i \rceil - 1$ and q_2 be $\Delta - C_i - p_2 T_i$, i.e., $p_2 + 1$ is the number of jobs of task τ_i that can be released in $[t_0 + C_i, t_d]$. Due to the assumption $\Delta > C_i$, we know that p_2 is a non-negative integer, $0 < q_2 \leq T_i$, and $\Delta - C_i = p_2 T_i + q_2$.

By the above definition, we achieve $\phi_i + \Delta = (p_1 + p_2) T_i + q_1 + q_2 + C_i$, and

$$
\begin{aligned}
& work_i(\Delta + \phi_i) - \rho\phi_i \\
& = work_i((p_1 + p_2)T_i + q_1 + q_2 + C_i) - \rho(p_1 T_i + q_1) \\
& = work_i(p_2 T_i + q_1 + q_2 + C_i) + p_1 C_i - \rho(p_1 T_i + q_1) \\
& = work_i(p_2 T_i + q_1 + q_2 + C_i) + p_1 U_i T_i - \rho(p_1 T_i + q_1) \\
& \leq work_i(p_2 T_i + q_1 + q_2 + C_i) - \rho q_1 \qquad (13)
\end{aligned}
$$

where the inequality is due to the assumption that $0 \leq U_i \leq \rho$. We will prove that the right-hand side of Eq. (9) is a safe upper bound on the condition in Eq. (13). By the definition of q_1 and q_2, we know that $0 \leq q_1 + q_2 \leq 2T_i$, i.e., $C_i \leq p_2 T_i + q_1 + q_2 + C_i \leq 2T_i + C_i$. Depending on the value of $q_1 + q_2$, there are four cases for different (linear or constant) segments of $work_i(p_2 T_i + q_1 + q_2 + C_i)$ to be analyzed:

- **Case 1:** $0 \leq q_1 + q_2 \leq T_i - C_i$: That is, $p_2 T_i + C_i \leq p_2 T_i + q_1 + q_2 + C_i \leq p_2 T_i + T_i$. Therefore, $work_i(p_2 T_i + C_i) \leq work_i(p_2 T_i + q_1 + q_2 + C_i) \leq work_i(p_2 T_i + T_i)$. Since $work_i(p_2 T_i + C_i) = work_i(p_2 T_i + T_i) = (p_2 + 1)C_i$, we have

$$
\text{RHS. of Eq. (13)} = (p_2 + 1)C_i - \rho q_1 \leq work_i(p_2 T_i + C_i + q_2) = work_i(\Delta),
$$

  where $\leq$ is due to $\rho \geq 0$ and $q_1 > 0$.
- **Case 2:** $T_i - C_i < q_1 + q_2 \leq T_i$: By definition, when p_2 is a nonnegative integer and $0 < x \leq C_i$, $work_i((p_2{+}1)T_i{+}x) = (p_2{+}1)C_i{+}x$. By $T_i{-}C_i < q_1{+}q_2 \leq T_i$, we know that $(p_2{+}1)T_i < p_2 T_i{+}q_1{+}q_2{+}C_i \leq (p_2{+}1)T_i{+}C_i$. Therefore, $work_i(p_2 T_i{+}q_1{+}q_2{+}C_i) = (p_2 + 1)C_i + (p_2 T_i + q_1 + q_2 + C_i - (p_2 + 1)T_i) = (p_2 + 1)C_i + (q_1 + q_2 + C_i - T_i)$. Let η be $T_i - (q_1 + q_2)$. By definition $\eta \geq 0$. Therefore,

$$
\begin{aligned}
\text{RHS. of Eq. (13)} = & (p_2 + 1)C_i + (C_i - \eta) - \rho(T_i - q_2 - \eta) \\
= & (p_2 + 1)C_i + (C_i - \rho(T_i - q_2)) + \eta(\rho - 1) \\
\leq & (p_2 + 1)C_i + \max\{0, C_i - \rho(T_i - q_2)\},
\end{aligned}
$$

  where $\leq$ is due to $0 \leq \rho \leq 1$ and $\eta \geq 0$.
- **Case 3:** $T_i < q_1 + q_2 \leq 2T_i - C_i$: Thus, $work_i(p_2 T_i + q_1 + q_2 + C_i) = (p_2 + 2)C_i$, and

$$
\text{RHS. of Eq. (13)} = (p_2 + 1)C_i + C_i - \rho q_1 \leq (p_2 + 1)C_i + \max\{0, C_i - \rho(T_i - q_2)\},
$$

  where $\leq$ is due to $\rho \geq 0$ and $q_1 + q_2 > T_i$.
- **Case 4:** $2T_i - C_i < q_1 + q_2 \leq 2T_i$: In this case $work_i(p_2 T_i + q_1 + q_2 + C_i)$ is equal to $(p_2{+}2)C_i{+}(q_1{+}q_2{+}C_i{-}2T_i)$, similar to the analysis in Case 2. Let η be $2T_i - (q_1{+}q_2)$. By definition $\eta \geq 0$. Therefore,

$$
\begin{aligned}
\text{RHS. of Eq. (13)} = & (p_2 + 1)C_i + 2C_i - \eta - \rho(2T_i - q_2 - \eta) \\
= & (p_2 + 1)C_i + C_i + T_i(U_i - \rho) - \eta(1 - \rho) - \rho(T_i - q_2) \\
\leq & (p_2 + 1)C_i + \max\{0, C_i - \rho(T_i - q_2)\},
\end{aligned}
$$

where $\leq$ is due to $0 < U_i \leq \frac{C'_k}{D'_k} \leq \rho \leq 1$ and $\eta \geq 0$, i.e., $U_i - \rho \leq 0$ and $-\eta(1-\rho) \leq 0$. Since $0 < q_1 + q_2 \leq 2T_i$, we know that $work_i(\Delta)$ is a safe upper bound for **Case 1** and that $(p_2+1)C_i + \max\{0, C_i - \rho(T_i - q_2)\}$ is a safe upper bound for the other cases, and we reach the conclusion of this lemma. ◀

▶ **Lemma 3.9.** *If $U_i \leq \rho$, then $\omega_i^{heavy}(\Delta) \geq \omega_i^{light}(\Delta)$ for all $\Delta > 0$.*

Proof. This inequality can be proved formally, but can also be derived by following the definitions. When $0 < \Delta \leq C_i$, the inequality holds naturally. In the proof of Lemma 3.8, *the workload of task τ_i that is executed in the time interval $[t_i, t_d)$ in schedule S is at most $work_i(t_d - t_i) = work_i(\Delta + \phi_i)$*. Since $\phi_i \leq D_i$, we know that $\omega_i^{light}(\Delta) \leq work_i(\Delta + \phi_i) \leq work_i(\Delta + D_i) = \omega_i^{heavy}(\Delta)$. ◀

Here is a short summary of the information provided by Lemmas 3.6, 3.7, and 3.8.

- According to Lemma 3.6, at time t_0, there are at most $\lceil M - (M-1)\rho \rceil - 1 = \lceil \mu_k \rceil - 1$ carry-in tasks.
- Among the $\lceil \mu_k \rceil - 1$ carry-in tasks, there are two types of carry-in tasks, i.e., *heavy* and *light* tasks. A light carry-in task τ_i can be described by $\omega_i^{light}(\Delta)$ from Eq. (9) if the utilization is no more than ρ and a heavy carry-in task τ_i can be described by $\omega_i^{heavy}(\Delta)$ from Eq. (8). By observing the conditions in Eqs. (8) and (9), we know that $work_i(\Delta) \leq \omega_i^{light}(\Delta) \leq \omega_i^{heavy}(\Delta)$.
- Since ρ is a user-defined parameter, a smaller ρ implies a larger μ_k, i.e., potentially more carry-in tasks and more heavy carry-in tasks. By constrast, a larger ρ implies a smaller μ_k, i.e., potentially less carry-in tasks and more light carry-in tasks. Therefore, *a larger ρ is better for minimizing the carry-in workload.*
- However, the window of interest $[t_0, t_d)$ is defined by the condition $\Omega(t_0, t_d) \geq M - (M-1)\rho$. The window of interest is smaller when ρ is larger. As a result, there is no monotonicity with respect to the schedulability test for setting the value of ρ.

▶ **Theorem 3.10.** *Task τ_k is schedulable by the given global fixed-priority scheduling if*

$$\forall \ell \in \mathbb{N}, \exists 1 \geq \rho \geq \ell C_k/((\ell-1)T_k + D_k), \forall \Delta \geq (\ell-1)T_k + D_k$$

$$\ell C_k + \sum_{\tau_i \in \mathbf{T}^{carry}} \omega_i^{diff}(\Delta, \rho) + \sum_{i=1}^{k-1} work_i(\Delta) \leq \Delta \cdot \mu_k \tag{14}$$

holds, where $\mu_k = M - (M-1)\rho$,

$$\omega_i^{diff}(\Delta, \rho) = \begin{cases} \omega_i^{heavy}(\Delta) - work_i(\Delta) & \text{if } U_i > \rho \\ \omega_i^{light}(\Delta) - work_i(\Delta) & \text{if } U_i \leq \rho \end{cases} \tag{15}$$

and $\mathbf{T}^{carry}$ is the set of the $\lceil \mu_k \rceil - 1$ tasks among the $k-1$ higher-priority tasks with the largest values of $\omega_i^{diff}(\Delta, \rho)$. If $D_k \leq T_k$, we only need to consider $\ell = 1$.

Proof. We prove this theorem by contrapositive, i.e., task τ_k misses its deadline first at time t_d in a global fixed-priority preemptive schedule S. We know that t_a can be defined for schedule S, and t_0, i.e., $\Omega(t_0, t_d) \geq M - (M-1) \times \frac{C'_k}{D'_k}$ in Definition 3.3 can be defined for any ρ with $1 \geq \rho \geq \ell C_k/((\ell-1)T_k + D_k)$ due to Lemma 3.4.

By the existence of t_d, the choice of ρ, and the definition of t_0 in Definition 3.3, we know that the deadline miss of task τ_k at time t_d in the schedule S implies

$$\exists \ell \in \mathbb{N}, \forall 1 \geq \rho \geq \ell C_k/((\ell-1)T_k + D_k), \exists \Delta = t_d - t_0, \qquad \Omega(t_0, t_d) \geq M - (M-1)\rho \tag{16}$$

By the fact that $C_k^* < C_k' = \ell C_k$ and the definition of $\Omega()$, we have

$$\Omega(t_0, t_d) = \frac{C_k^* + E(t_0, t_d)}{t_d - t_0} < \frac{\ell C_k + E(t_0, t_d)}{t_d - t_0} \tag{17}$$

By Lemma 3.6, for a specific ρ, there are at most $\lceil M - (M-1)\rho \rceil - 1 = \lceil \mu_k \rceil - 1$ higher-priority carry-in tasks at time t_0 and the other higher-priority tasks do not have any unfinished job at time t_0. Suppose that $\mathbf{T}^{heavy}$ and $\mathbf{T}^{light}$ are the sets of the heavy and light carry-in tasks at time t_0, respectively. By Lemma 3.6, $|\mathbf{T}^{heavy}| + |\mathbf{T}^{light}| \leq \lceil \mu_k \rceil - 1$. Therefore, by using Lemmas 3.7 and 3.8 and 3.9, we have

$$E(t_0, t_d) \leq \sum_{\tau_i \in \mathbf{T}^{heavy}} \omega_i^{heavy}(\Delta) + \sum_{\tau_i \in \mathbf{T}^{light}} \omega_i^{light}(\Delta)$$

$$= \sum_{\tau_i \in \mathbf{T}^{heavy}} \left(\omega_i^{heavy}(\Delta) - work_i(\Delta)\right) + \sum_{\tau_i \in \mathbf{T}^{light}} \left(\omega_i^{light}(\Delta) - work_i(\Delta)\right) + \sum_{i=1}^{k-1} work_i(\Delta) \tag{18}$$

$$\leq \sum_{\tau_i \in \mathbf{T}^{carry}} \omega_i^{diff}(\Delta, \rho) + \sum_{i=1}^{k-1} work_i(\Delta) \tag{19}$$

where $\omega_i^{diff}(\Delta, \rho)$ is defined in Eq. (15), and $\mathbf{T}^{carry}$ is defined in the statement of the theorem.

By Eqs. (16), (17), and (19), and the fact $t_d - t_a \geq D_k' = (\ell - 1)T_k + D_k$, the deadline miss of task τ_k at t_d implies

$$\exists \ell \in \mathbb{N}, \forall 1 \geq \rho \geq \ell C_k / ((\ell - 1)T_k + D_k), \exists \Delta \geq (\ell - 1)T_k + D_k$$

$$\ell C_k + \sum_{\tau_i \in \mathbf{T}^{carry}} \omega_i^{diff}(\Delta, \rho) + \sum_{i=1}^{k-1} work_i(\Delta) > \Delta \cdot \mu_k \tag{20}$$

Therefore, the negation of the above necessary condition for the deadline miss of task τ_k at time t_d is a safe sufficient schedulability test. We reach the conclusion of the schedulability test.

When $D_k \leq T_k$, since t_d is the earliest moment in the schedule S with a deadline miss of task τ_k, we know that t_a is by definition $t_d - D_k$ and ℓ is 1. Therefore, we only have to consider $\ell = 1$ when $D_k \leq T_k$. ◀

The schedulability test described in Theorem 3.10 can be informally explained as follows: 1) it requires to test all the possible positive integers for ℓ, like the busy-window concept, 2) it has to find a ρ value in the specified range, and 3) for the specified combination of ℓ and ρ, we have to test whether the condition in Eq. (14) holds for every $\Delta \geq (\ell - 1)T_k + D_k$.

3.3 Remarks on Implementing Theorem 3.10

Unfortunately, due to the following issues, implementing the schedulability test in Theorem 3.10 directly would lead to a high time complexity:

- **Issue 1 due to Δ**: For specific ℓ and ρ, testing the schedulability condition in Eq. (14) requires to evaluate all $\Delta \geq (\ell - 1)T_k + D_k$. Suppose that $HP(k)$ is the hyper-period of $\{\tau_1, \tau_2, \ldots, \tau_{k-1}\}$, i.e., the least common multiple of the periods of $\tau_1, \tau_2, \ldots, \tau_{k-1}$. Since $work_i(\Delta) + HP(k)U_i = work_i(\Delta + HP(k))$, $\omega_i^{light}(\Delta) + HP(k)U_i = \omega_i^{light}(\Delta + HP(k))$, and $\omega_i^{heavy}(\Delta) + HP(k)U_i = \omega_i^{heavy}(\Delta + HP(k))$, we only have to test $\Delta \in [(\ell-1)T_k + D_k, (\ell - 1)T_k + D_k + HP(k)]$, as long as $\sum_{i=1}^{k-1} U_i \leq \mu_k$. However, the time complexity can still be exponential. We will explain how to reduce this complexity by using safe upper bounds in Section 4.

- **Issue 2 due to ρ**: For a specific ℓ, the schedulability condition in Eq. (14) is dependent on the selection of ρ. If ρ is smaller, then μ_k is larger, and vice versa. A smaller ρ increases the right-hand side in the schedulability test in Eq. (14), but it also increases the left-hand side, since there are potentially more carry-in tasks. One simple strategy to find a suitable ρ instead of searching for all values of ρ is to start from $\rho = \ell C_k/((\ell-1)T_k + D_k$ and increase ρ to the next (higher) U_i for certain higher-priority task τ_i if necessary. Therefore, in the worst case, we only have to consider k different ρ values. We will deal with this in Theorems 4.4 and 4.5 in Section 4.
- **Issue 3 due to ℓ**: We need to consider all positive integer values of ℓ in the schedulability condition in Eq. (14), as the test is only valid when the condition holds for all $\ell \in \mathbb{N}$. Therefore, if we only test some ℓ, it is necessary to prove that the other ℓ configurations are also covered even though they are not tested. We will explain how to deal with this in Theorems 4.6 and 4.7 in Section 4.

4 Efficient Schedulability Tests

In this section we provide several schedulability tests based on approximate workload functions to test the schedulability of task τ_k more efficiently. The following three lemmas approximate the *piecewise linear* workload function $work_i(\Delta)$, $\omega_i^{heavy}(\Delta)$ and $\omega_i^{light}(\Delta)$ by *linear* functions with respect to Δ for any $\Delta \geq 0$.

▶ **Lemma 4.1.** *When $0 \leq U_i \leq 1$, for any $\Delta \geq 0$,*

$$work_i(\Delta) \leq C_i - C_iU_i + U_i\Delta. \tag{21}$$

Proof. This inequality was already stated in Eq. (5) by Bini et al. [14] as a fact. Here, we provide the proof for completeness. Suppose that Δ is $p_3T_i + q_3$, where p_3 is $\left\lfloor \frac{\Delta}{T_i} \right\rfloor$ and q_3 is $\Delta - \left\lfloor \frac{\Delta}{T_i} \right\rfloor T_i$. Therefore, we know $U_i\Delta = p_3C_i + q_3U_i$ and $work_i(\Delta) = p_3C_i + \min\{C_i, q_3\}$. We have to consider two cases:

- If $q_3 \leq C_i$: we have

$$\begin{aligned} work_i(\Delta) = {} & p_3C_i + q_3 \leq p_3C_i + C_i - (C_i - q_3) \\ \leq_1 {} & p_3C_i + C_i - (C_i - q_3)U_i = C_i - C_iU_i + U_i\Delta, \end{aligned}$$

  where $\leq_1$ is due to $0 \leq U_i \leq 1$ and $C_i - q_3 \geq 0$.
- If $q_3 > C_i$: we have

$$work_i(\Delta) = p_3C_i + C_i \leq p_3C_i + C_i + (q_3 - C_i)U_i = C_i - C_iU_i + U_i\Delta,$$

  where $\leq$ is due to $0 \leq U_i \leq 1$ and $q_3 - C_i > 0$.

◀

▶ **Lemma 4.2.** *For any $\Delta \geq 0$,*

$$\omega_i^{heavy}(\Delta) \leq C_i + U_iD_i - C_iU_i + U_i\Delta. \tag{22}$$

Proof. Due to Lemma 3.7 and Lemma 4.1, the inequality holds. ◀

▶ **Lemma 4.3.** *If $U_i \leq \rho \leq 1$, for any $\Delta \geq 0$,*

$$\omega_i^{light}(\Delta) \leq C_i - C_iU_i + U_i\Delta. \tag{23}$$

Proof. We consider the three upper bounds in Lemma 3.8 individually. When $\Delta \leq C_i$, this follows from Lemma 4.1 directly. When $\Delta > C_i$ and $\omega_i^{light}(\Delta) = work_i(\Delta)$, it holds due to Lemma 4.1 as well.

For the last case we have to bound $(p_2+1)C_i + \max\{0, C_i - \rho(T_i - q_2)\}$, as defined in Lemma 3.8. By the definition of p_2 and q_2, i.e., $\Delta - C_i = p_2 T_i + q_2$, in the statement of Lemma 3.8, we have $p_2 + 1 = \lceil (\Delta - C_i)/T_i \rceil$ and $(p_2+1)C_i = work_i(p_2 T_i + C_i) = work_i(\Delta - q_2)$. Therefore, for any $\Delta > C_i$, if $C_i - \rho(T_i - q_2) \geq 0$, we get

$$\begin{aligned}
\omega_i^{light}(\Delta) =&\ (p_2+1)C_i + C_i - \rho(T_i - q_2) \\
=&\ work_i(\Delta - q_2) + C_i - \rho(T_i - q_2) \\
\leq_1&\ C_i - C_i U_i + U_i(\Delta - q_2) + C_i - \rho(T_i - q_2) \\
=&\ C_i - C_i U_i + U_i \Delta - q_2(U_i - \rho) - T_i(\rho - U_i) \\
=&\ C_i - C_i U_i + U_i \Delta + (T_i - q_2)(U_i - \rho) \\
\leq_2&\ C_i - C_i U_i + U_i \Delta,
\end{aligned}$$

where $\leq_1$ is due to Lemma 4.1 and $\leq_2$ is due to $q_2 \leq T_i$ and $U_i \leq \rho$. For any $\Delta > C_i$, if $C_i - \rho(T_i - q_2) < 0$, similarly, we have

$$\begin{aligned}
\omega_i^{light}(\Delta) =&(p_2+1)C_i = work_i(p_2 T_i + C_i) \leq work_i(p_2 T_i + C_i + q_2) = work_i(\Delta) \\
\leq& C_i - C_i U_i + U_i \Delta.
\end{aligned}$$

Therefore, we reach the conclusion. ◀

With the help of the above lemmas for safe approximations, we can now safely and efficiently handle the schedulability test for specific ℓ and ρ in the following theorem. This handles **Issue 1** explained at the end of Section 3.

▶ **Theorem 4.4.** *Task τ_k is schedulable by the given global fixed-priority scheduling if*

$$\forall \ell \in \mathbb{N}, \exists 1 \geq \rho \geq \ell C_k/((\ell-1)T_k + D_k)$$

$$\frac{\ell C_k}{D'_k} + \sum_{\tau_i \in \mathbf{T}^{carry-approx}} \frac{\gamma_i U_i D_i}{D'_k} + \sum_{i=1}^{k-1} \left(\frac{C_i - C_i U_i}{D'_k} + U_i \right) \leq \mu_k, \tag{24}$$

where $\mu_k = M - (M-1)\rho$ with $1 \geq \rho \geq \ell C_k/((\ell-1)T_k + D_k)$, D'_k is $(\ell-1)T_k + D_k$,

$$\gamma_i = \begin{cases} 1 & \text{if } U_i > \rho \\ 0 & \text{if } U_i \leq \rho \end{cases} \tag{25}$$

and $\mathbf{T}^{carry-approx}$ is the set of the $\lceil \mu_k \rceil - 1$ tasks among the $k-1$ higher-priority tasks with the largest values of $\gamma_i U_i D_i$. Note that $|\mathbf{T}^{carry-approx}|$ can be smaller than $\lceil \mu_k \rceil - 1$ if the number of tasks with $U_i > \rho$ is less than $\lceil \mu_k \rceil - 1$. If $D_k \leq T_k$, we only need to consider $\ell = 1$.

Proof. We prove that the condition in this theorem is a safe upper bound of that in Theorem 3.10. For specific ℓ, ρ, Δ, we can find $\mathbf{T}^{carry}$ as defined in Theorem 3.10. By Lemmas 4.1, 4.2, and 4.3 and the assumptions $\Delta \geq (\ell-1)T_k + D_k = D'_k$ and $0 < U_i \leq 1 \forall \tau_i$, we have

$$\ell C_k + \sum_{\tau_i \in \mathbf{T}^{carry}} \omega_i^{diff}(\Delta, \rho) + \sum_{i=1}^{k-1} work_i(\Delta)$$

$$\leq \ell C_k + \sum_{\tau_i \in \mathbf{T}^{carry}} \gamma_i U_i D_i + \sum_{i=1}^{k-1} (C_i - C_i U_i + U_i \Delta) \tag{26}$$

$$\leq \ell C_k + \sum_{\tau_i \in \mathbf{T}^{carry-approx}} \gamma_i U_i D_i + \sum_{i=1}^{k-1} (C_i - C_i U_i + U_i \Delta) \tag{27}$$

$$\leq \Delta \cdot \left(\frac{\ell C_k}{D'_k} + \sum_{\tau_i \in \mathbf{T}^{carry-approx}} \frac{\gamma_i U_i D_i}{D'_k} + \sum_{i=1}^{k-1} \left(\frac{C_i - C_i U_i}{D'_k} + U_i \right) \right) \tag{28}$$

Therefore, the test in Theorem 3.10 can be safely over-approximated as follows:

$$\forall \ell \in \mathbb{N}, \exists 1 \geq \rho \geq \ell C_k/((\ell-1)T_k + D_k)T_k + D_k$$

$$\frac{\ell C_k}{D'_k} + \left(\sum_{\tau_i \in \mathbf{T}^{carry-approx}} \frac{\gamma_i U_i D_i}{D'_k} \right) + \sum_{i=1}^{k-1} \left(\frac{C_i - C_i U_i}{D'_k} + U_i \right) \leq \mu_k \tag{29}$$

◀

Theorem 4.4 provides two interesting implications to handle **Issue 2**. Firstly, if $U_i \leq \rho$, the linear approximation of $work_i(\Delta)$ by considering task τ_i as a non-carry-in task in Lemma 4.1 is the same as the linear approximation of $\omega_i^{light}(\Delta)$ by considering task τ_i as a carry-in task in Lemma 4.3. Therefore, the carry-in tasks are only effective for those tasks τ_i with $U_i > \rho$. Secondly, for a specific ℓ, deciding whether a specific ρ exists to pass the test in Eq. (24) can be done by only testing a finite number of ρ values, i.e. by starting from $\rho = \ell C_k/((\ell-1)T_k + D_k$ and increasing ρ to the next (higher) values where $\mathbf{T}^{carry-approx}$ changes. This means either 1) $\rho = U_i$ for certain higher-priority task τ_i, i.e., the summation can be larger with the same number of summands; or 2) $\mu_k = M-(M-1)\rho$ is an integer, i.e., the number of summands increases. This only has time complexity $O((k+M)\log(k+M))$, mainly due to the sorting, when proper data structures are used. Details can be found in Appendix of the full version [21].

4.1 Linear-Time Schedulability Tests

The time complexity of Theorem 4.4 is due to the search of possible ρ values. Nevertheless, we can directly set ρ to $U^{\max}_{\delta,k}$ which implies that there is no carry-in task in the linear-approximation form. With this simplification, we can conclude different schedulability tests in Theorems 4.5, 4.6, and 4.7. Although these tests are not superior to Theorem 4.4, our main target is the test in Theorem 4.7, which will be used *mainly to derive the speedup bounds later in Theorem 5.2.*

▶ **Theorem 4.5.** *Task τ_k is schedulable by the given global fixed-priority scheduling if* $\forall \ell \in \mathbb{N}$

$$\frac{\ell C_k}{D'_k} + \sum_{i=1}^{k-1} \left(\frac{C_i - C_i U_i}{D'_k} + U_i \right) \leq (M - (M-1) U^{\max}_{\delta,k}) \tag{30}$$

holds, where D'_k is $(\ell-1)T_k + D_k$.

Proof. This comes directly from Theorem 4.4 by setting ρ to $U^{\max}_{\delta,k}$ and the facts that $U^{\max}_{\delta,k} \geq U_i$ for $i = 1, 2, \ldots, k-1$ and $U^{\max}_{\delta,k} \geq \delta_k \geq \ell C_k/((\ell-1)T_k + D_k)$ by definition. ◀

▶ **Theorem 4.6.** *Suppose that $D_k > T_k$. Let b be $\frac{D_k - T_k}{T_k}$. Task τ_k is schedulable by the given global fixed-priority scheduling algorithm if:*

$$\sum_{i=1}^{k} U_i \leq (M - (M-1)U^{\max}_{\delta,k}), \qquad \text{when } bU_k - \sum_{i=1}^{k-1} \frac{C_i - C_i U_i}{T_k} > 0 \tag{31}$$

$$\frac{C_k}{D_k} + \sum_{i=1}^{k-1} (\frac{C_i - C_i U_i}{D_k} + U_i) \leq (M - (M-1)U^{\max}_{\delta,k}), \qquad \text{otherwise} \tag{32}$$

Proof. For a given ℓ, the left-hand side in Eq. (30) can be rephrased as:

$$F(\ell) = \frac{\ell C_k}{D'_k} + \sum_{i=1}^{k-1} \left(\frac{C_i - C_i U_i}{D'_k} + U_i \right) = \frac{\ell U_k + \sum_{i=1}^{k-1} \frac{C_i - C_i U_i}{T_k}}{\ell + b} + \sum_{i=1}^{k-1} U_i \tag{33}$$

The first order derivative of $F(\ell)$ with respect to ℓ is:

$$\frac{\partial F(\ell)}{\partial \ell} = \frac{bU_k - \sum_{i=1}^{k-1} \frac{C_i - C_i U_i}{T_k}}{(\ell + b)^2}. \tag{34}$$

We have to consider two cases:

- Case 1: if $bU_k - \sum_{i=1}^{k-1} \frac{C_i - C_i U_i}{T_k} > 0$, then $F(\ell)$ is an increasing function with respect to ℓ. Therefore, $F(\ell)$ is maximized when $\ell \to \infty$, i.e., $F(\ell) \leq \sum_{i=1}^{k} U_i$.
- Case 2: if $bU_k - \sum_{i=1}^{k-1} \frac{C_i - C_i U_i}{T_k} \leq 0$, then $F(\ell)$ is a non-increasing function with respect to ℓ. Therefore, $F(\ell)$ is maximized when $\ell \to 1$, i.e., $F(\ell) \leq \frac{C_k}{D_k} + \sum_{i=1}^{k-1} (\frac{C_i - C_i U_i}{D_k} + U_i)$.

◀

▶ **Theorem 4.7.** *Task τ_k is schedulable by the given global fixed-priority scheduling if*

$$\delta_k + \sum_{i=1}^{k-1} \left(\frac{C_i - C_i U_i}{D_k} + U_i \right) \leq M - (M-1)U^{\max}_{\delta,k} \tag{35}$$

Proof. Based on Theorem 4.5 and the two facts that $D'_k = (\ell-1)T_k + D_k \geq D_k$ and $\delta_k \geq \ell C_k/((\ell-1)T_k + D_k)$ for all $\ell \in \mathbb{N}$, we reach the conclusion. ◀

4.2 Dominance

We now show analytical dominance among the tests presented above and in Theorem 3.10 in the following corollary. A test $\mathcal{B}_1$ analytically dominates another test $\mathcal{B}_2$ if the schedulability condition in $\mathcal{B}_1$ always dominates that in $\mathcal{B}_2$. This means, if task τ_k is deemed schedulable by $\mathcal{B}_2$, task τ_k is also deemed schedulable by $\mathcal{B}_1$.

▶ **Corollary 4.8.** *For arbitrary-deadline sporadic real-time systems under global fixed-priority scheduling, the schedulability tests have the following dominance relations.*

- *Theorem 3.10 analytically dominates Theorem 4.4.*
- *Theorem 4.4 analytically dominates Theorem 4.5.*
- *Theorem 4.5 is equivalent to the test in Theorem 4.6.*
- *Theorem 4.6 analytically dominates Theorem 4.7.*

Proof. They follow directly from the above analyses. The reason why Theorems 4.5 and 4.6 are equivalent is because the conditions in Theorem 4.6 represent exactly the worst-case ℓ selection in Theorem 4.6. The other cases are obvious. ◀

Although we will show in Theorem 5.3 that all the above schedulability tests have the same speedup bound for global DM, the *performance* of the schedulability tests in this section can be very different *in practice*. Chen et al. [19] have recently shown that "*Speedup factors … often lack the power to discriminate between the performance of different scheduling algorithms and schedulability tests even though the performance of these algorithms and tests may be very different when viewed from the perspective of empirical evaluation.*" To avoid concluding an algorithm with a reasonable speedup bound but practically not useful, we performed a series of experiments and present the results in Section 6. Moreover, according to the experimental results, the domination relations among Theorems 4.4, 4.5, and 4.7 are strict, i.e., there is a concrete input instance that is deemed schedulable by a dominating schedulability test but is not deemed schedulable by a dominated schedulability test.

5 Global Deadline-Monotonic (DM) Scheduling

After presenting the schedulability tests for any global fixed-priority scheduling algorithms, we focus ourselves on global DM in this section. We will discuss the speedup upper bound and the speedup lower bound. Baruah and Fisher [8] showed that global DM has a speedup upper bound of $2 + \sqrt{3} \approx 3.73$ compared to the optimal schedules, based on the test restated in Theorem 2.4. This is the best known upper bound on speedup factors for arbitrary-deadline sporadic task systems under global fixed-priority scheduling. Evaluating $\text{LOAD}(k)$ in Theorem 2.4 requires to calculate $\sum_{i=1}^{k} \text{DBF}(\tau_i, t)/t$ at all time points t. This means, the naïve implementation has an exponential-time complexity. There are more efficient methods, as discussed by Baruah and Bini [6], but the time complexity remains exponential. Although it is possible to approximate $\text{LOAD}(k)$ by using approximate demand bound functions in polynomial time, this is at a price of higher $\text{LOAD}(k)$. We show that the test in Theorem 2.4 is over-pessimistic and is analytically dominated by our linear-time schedulability test in Theorem 4.7 under global DM.

▶ **Corollary 5.1.** *For global DM, the schedulability test in Theorem 4.7 analytically dominates the schedulability test in Theorem 2.4 proposed by Baruah and Fisher [8].*

Proof. This is due to the following facts:

- By definition, $\text{LOAD}(k) \geq \text{limit}_{t\to\infty} \sum_{i=1}^{k} \text{DBF}(\tau_i, t)/t = \sum_{i=1}^{k} U_i$.
- Since $D_i \leq D_k$ in global DM for $i = 1, 2, \ldots, k-1$, we know that $\frac{\sum_{i=1}^{k} \text{DBF}(\tau_i, D_k)}{D_k} \geq \sum_{i=1}^{k} \frac{C_i}{D_k}$. Therefore, $\text{LOAD}(k) \geq \sum_{i=1}^{k} \frac{C_i}{D_k}$.

Combining these facts, we get

$$\delta_k + \sum_{i=1}^{k-1} \left(\frac{C_i - C_i U_i}{D_k} + U_i \right) \leq \sum_{i=1}^{k} \frac{C_i}{D_k} + U_i \leq 2\text{LOAD}(k). \tag{36}$$

Since we know that the right-hand side in Eq. (4), i.e., $M - (M-1)\delta_{\max}(k)$, is less than or equal to $M - (M-1)U_{\delta,k}^{\max}$ in Eq. (35), we reach the conclusion. ◀

▶ **Theorem 5.2.** *Global DM has a speedup bound of $3 - \frac{1}{M}$, with respect to the optimal schedule, when $M \geq 2$.*

Proof. We only prove the speedup bound by using the schedulability test in Theorem 4.7. Due to the dominance properties in Corollary 4.8, such a bound also holds for the schedulability tests from Theorems 3.10, 4.4, 4.5, and 4.6.

Suppose that task τ_k is not schedulable by global DM. Since $D_i \leq D_k$ for any $i = 1, 2, \ldots, k-1$ under global DM, we know $\text{DBF}(\tau_i, D_k) \geq C_i$. Therefore, under global DM, $\sum_{i=1}^{k} \frac{C_i}{MD_k} \leq \sum_{i=1}^{k} \frac{\text{DBF}(\tau_i, D_k)}{MD_k} \leq \frac{\sum_{\tau_i \in \mathbf{T}} \text{DBF}(\tau_i, D_k)}{MD_k} \leq \max_{t>0} \frac{\sum_{\tau_i \in \mathbf{T}} \text{DBF}(\tau_i, t)}{Mt}$.

By the assumption that task τ_k is also deemed not schedulable by Theorem 4.7, we have

$$
\begin{aligned}
& & \delta_k + \sum_{i=1}^{k-1} \left(\frac{C_i - C_i U_i}{D_k} + U_i \right) > M - (M-1) U_{\delta,k}^{\max} \\
& \Rightarrow & \sum_{i=1}^{k} \frac{C_i}{MD_k} + \sum_{i=1}^{k} \frac{U_i}{M} > 1 - \left(1 - \frac{1}{M}\right) U_{\delta,k}^{\max} \\
& \Rightarrow & \sum_{i=1}^{k} \frac{C_i}{MD_k} + \sum_{i=1}^{k} \frac{U_i}{M} + \left(1 - \frac{1}{M}\right) U_{\delta,k}^{\max} > 1 \\
& \xRightarrow{x + x + (1-1/M)x > 1} & \max \left\{ \sum_{i=1}^{k} \frac{C_i}{MD_k}, \sum_{i=1}^{k} \frac{U_i}{M}, U_{\delta,k}^{\max} \right\} > \frac{1}{3 - 1/M}
\end{aligned} \tag{37}
$$

Therefore, either $\max_{t>0} \frac{\sum_{\tau_i \in \mathbf{T}} \text{DBF}(\tau_i, t)}{Mt} \geq \sum_{i=1}^{k} \frac{C_i}{MD_k} > \frac{1}{3-1/M}$, or $\sum_{i=1}^{k} \frac{U_i}{M} > \frac{1}{3-1/M}$, or $\delta_{\max}(k) \geq U_{\delta,k}^{\max} > \frac{1}{3-1/M}$. By Lemma 2.3, we reach the conclusion of the speedup bound for global DM with respect to the optimal schedule. ◀

▶ **Theorem 5.3.** *For global DM, the schedulability tests in Theorems 3.10, 4.4, 4.5, 4.6, and 4.7 have a speedup bound of* $3 - \frac{1}{M}$*, with respect to the optimal schedule, when* $M \geq 2$.

Proof. This is due to Theorem 5.2 and Corollary 4.8, because all of the tests in Theorems 3.10, 4.4, 4.5, 4.6 dominate the test in Theorem 4.7 as presented in Corollary 4.8. ◀

▶ **Theorem 5.4.** *The speedup bound of global DM for arbitrary-deadline task systems is at least* $3 - \frac{3}{M+1}$.

Proof. The proof is based on a concrete task set. We specifically use the following task set $\mathbf{T}^{ad}$ with $N = 2M + 1$ tasks. Let ε be an arbitrarily small positive real number such that $1/\varepsilon$ is an integer. Let $\eta \ll \varepsilon$ be an arbitrarily small positive number, that is used to enforce the priority assignment under global DM:

- $C_i = \frac{\varepsilon}{3}$, $T_i = \varepsilon$, $D_i = 1$, for $i = 1, 2, \ldots, M$.
- $C_i = \frac{1}{3}$, $T_i = \infty$, $D_i = 1 + \eta$, for $i = M+1, M+2, \ldots, 2M$.
- $C_i = \frac{1+\varepsilon}{3}$, $T_i = \infty$, $D_i = 1 + 2\eta$, for $i = 2M + 1$

As the setting of $\eta \ll \varepsilon$ is just to enforce the indexing, *we will directly take* $\eta \rightarrow 0$ *here.* In the Appendix, we prove two properties: 1) $\mathbf{T}^{ad}$ is not schedulable by global DM under a concrete instance which releases all the tasks at time 0 and the subsequent jobs periodically. 2) There exists a feasible schedule for task set $\mathbf{T}^{ad}$ at any speed no lower than $\frac{1+\varepsilon}{3} + \frac{1+\varepsilon}{3M}$ under a concrete semi-partitioned multiprocessor schedule, i.e., $\{\tau_m, \tau_{m+M}\}$ assigned to processor m for $m = 1, 2, \ldots, M$ and task τ_{2M+1} executed partially on each of the M processors. Therefore, a lower bound on the speedup bound of global DM is:

$$
\lim_{\varepsilon \rightarrow 0} \frac{1}{\frac{1+\varepsilon}{3} + \frac{1+\varepsilon}{3M}} = \lim_{\varepsilon \rightarrow 0} \frac{3M}{(1+\varepsilon) \times (M+1)} = \frac{3M}{M+1} = 3 - \frac{3}{M+1}.
$$

◀

By Theorems 5.2 and 5.4, we can reach the conclusion that all the schedulability tests from Theorems 3.10, 4.4, 4.5, 4.6, and 4.7 are asymptotically tight with respect to speedup bounds. However, these tests have different performance with respect to the schedulability.

6 Evaluation

We evaluated the scheduling tests provided in this paper by comparing their acceptance ratio to the acceptance ratio of other algorithms, i.e., comparing the percentage of task sets accepted for the different schedulability tests, using different settings for the number of processors, the scheduling policy, and the ratio of the relative deadline to the period.

Evaluation Setup: We conducted evaluations for homogeneous multiprocessor systems with $M = 4$, $M = 8$, and $M = 16$ processors. We generated 100 task sets with cardinality of both $N = 5 \times M$ and $N = 10 \times M$, and utilization ranging from $M \times 5\%$ to $M \times 100\%$ in steps of $M \times 5\%$. The UUniFast-Discard method [13] was adopted to generate the utilization values of a set of N tasks under the target utilization. As suggested by Emberson et al. [28], the periods were generated according to a log-uniform distribution, with 1, 2, and 3 orders of magnitude, i.e., $[1ms - 10ms]$, $[1ms - 100ms]$, and $[1ms - 1000ms]$. For each task, the relative deadline was set to the period multiplied with a value randomly drawn under a uniform distribution from a given interval I. We conducted evaluations using different interval, i.e., I was $[0.8, 2], [0.8, 5], [0.8, 10], [1, 2], [1, 5]$, or $[1, 10]$. To schedule the task sets, we applied global deadline-monotonic (DM) and global slack-monotonic (SM) [1] scheduling.

Whether the task set is schedulable under the given scheduling approach or not was tested using the following schedulability tests:

- LOAD: The load-based analysis by Baruah and Fisher in [9], only for DM scheduling.
- BAK: The test by Baker in Theorem 11 in [3].
- HC: The sufficient test in Corollary 2 by Huang and Chen in [31].
- OUR-4.4: The sufficient test in Theorem 4.4 in this paper.
- OUR-4.6: The sufficient test in Theorem 4.6 in this paper.
- OUR-4.7: The sufficient test in Theorem 4.7 in this paper.

We also checked if a task set was schedulable according to at least one of the tests, denoted as *ALL. We only present a small set of the conducted tests here. The diagrams of all conducted evaluations can be found in [20].*

Evaluation Results: Figure 3 shows the evaluations under the setting used in the paper by Huang and Chen [31]. They used DM scheduling on $M = 8$ processors, a task set containing 40 tasks and ratios of $\frac{D_i}{T_i} \in [0.8, 2]$ and analyzed the schedulability for T_i values that differ up to 1, 2, and 3 orders of magnitude, i.e., T_i in a range of $[1ms, 10ms]$, $[1ms, 100ms]$, or $[1ms, 1000ms]$. The test by Baruah and Fisher [9] is clearly outperformed by Theorem 4.6, Theorem 4.7, and Baker's test [3], which provide similar acceptance ratios. The test by Huang and Chen [31] outperforms those three tests and is worse than the test in Theorem 4.4 in these settings. However, there is no dominance relation between Theorem 4.4 and the test by Huang and Chen [31], as some task sets are schedulable under the test by Huang and Chen [31] but not schedulable under Theorem 4.4 and vise versa.

There are other configurations where the test by Huang and Chen [31] performs better than Theorem 4.4. One example is shown in Figure 4, analyzing the impact of the number of processors. Here Theorem 4.4 performs compatible to Theorem 4.6, Theorem 4.7, and Baker's test [3] for $M = 4$. When the number of processors increases, Theorem 4.4 performs better. The gap to Huang and Chen [31] is smaller for 8 processors and Theorem 4.4 has a higher acceptance rate when the utilization level is $80\% \times M$. For $M = 16$ processors

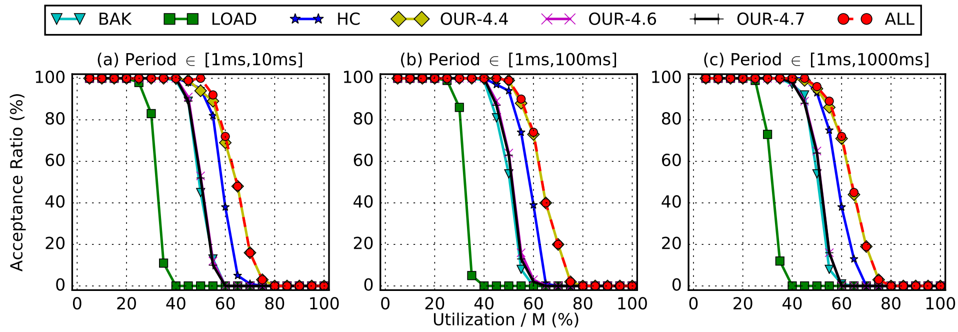


Figure 3 Comparison of the tests presented in Theorem 4.4, 4.6, and 4.7 with the methods from Baruah and Fisher (LOAD) [9], Baker [3], and Huang and Chen [31] for different ranges of period. The evaluation setup is the same as in [31], i.e., DM, $M = 8$, $N = 40$, $\frac{D_i}{T_i} \in [0.8, 2]$.

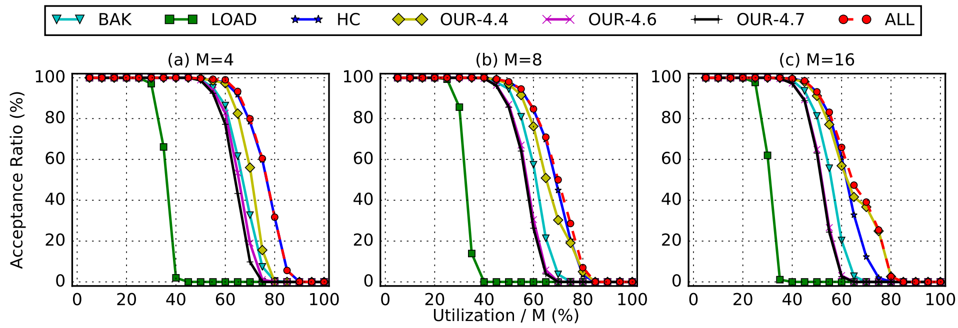


Figure 4 Comparison of the tests presented in Theorem 4.4, 4.6, and 4.7 with the methods from Baruah and Fisher (LOAD) [9], Baker [3], and Huang and Chen [31] for different M values. The other parameters are fixed, i.e., DM, $N = 5 \times M$, $T_i \in [1ms, 10ms]$, and $\frac{D_i}{T_i} \in [0.8, 10]$.

Theorem 4.4 accepts more task sets than Huang and Chen [31] when the utilization level is $\geq 65\% \times M$. In addition, it is possible that the number of task sets that is accepted by at least one algorithm is not close to the number of task sets accepted by Huang and Chen [31] or Theorem 4.4 as can be seen for the utilization level $75\% \times M$ in the case where $M = 8$.

Furthermore, we tracked if the test by Baker [3] accepted some task sets that were not accepted by Huang and Chen [31] or Theorem 4.4, which happened occasionally. Therefore, we conclude that there is no dominance relation between any of those three tests, i.e., Theorem 4.4, and the tests by Baker [3] and by Huang and Chen [31]. As these tests can all be implemented with polynomial-time complexity, all three should be applied.

7 Conclusion

We present a series of schedulability tests for multiprocessor systems under any given fixed-priority scheduling approach. Those schedulability tests have different tradeoffs between their accuracy and their time complexity. All those schedulability tests dominate the approach by Baruah and Fisher [9], both with respect to speedup bounds and schedulability analysis. Theorem 3.10 is the most powerful schedulability test in this paper. However, we do not reach any concrete implementation with affordable time complexity. In the future work, we will seek for efficient methods to implement the schedulability test in Theorem 3.10.

References

1 Björn Andersson. Global static-priority preemptive multiprocessor scheduling with utilization bound 38%. In *Principles of Distributed Systems, 12th International Conference, OPODIS*, pages 73–88, 2008. doi:10.1007/978-3-540-92221-6_7.

2 Björn Andersson, Sanjoy K. Baruah, and Jan Jonsson. Static-priority scheduling on multiprocessors. In *Real-Time Systems Symposium (RTSS)*, pages 193–202, 2001. doi:10.1109/REAL.2001.990610.

3 Theodore P Baker. An analysis of fixed-priority schedulability on a multiprocessor. *Real-Time Systems*, 32(1-2):49–71, 2006. doi:10.1007/S11241-005-4686-1.

4 Theodore P. Baker and Michele Cirinei. Brute-force determination of multiprocessor schedulability for sets of sporadic hard-deadline tasks. In *Principles of Distributed Systems, 11th International Conference, OPODIS*, pages 62–75, 2007. doi:10.1007/978-3-540-77096-1_5.

5 Sanjoy Baruah. Techniques for multiprocessor global schedulability analysis. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium*, pages 119–128, 2007. doi:10.1109/RTSS.2007.35.

6 Sanjoy Baruah and Enrico Bini. Partitioned scheduling of sporadic task systems: an ILP-based approach. In *Proc. DASIP*, 2008.

7 Sanjoy K. Baruah, Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, and Sebastian Stiller. Improved multiprocessor global schedulability analysis. *Real-Time Systems*, 46(1):3–24, 2010. doi:10.1007/s11241-010-9096-3.

8 Sanjoy K. Baruah and Nathan Fisher. Global deadline-monotonic scheduling of arbitrary-deadline sporadic task systems. In *Principles of Distributed Systems, 11th International Conference, OPODIS 2007, Guadeloupe, French West Indies, December 17-20, 2007. Proceedings*, pages 204–216, 2007. doi:10.1007/978-3-540-77096-1_15.

9 Sanjoy K. Baruah and Nathan Fisher. Global fixed-priority scheduling of arbitrary-deadline sporadic task systems. In *Distributed Computing and Networking, 9th International Conference, ICDCN*, pages 215–226, 2008. doi:10.1007/978-3-540-77444-0_20.

10 Sanjoy K. Baruah, Aloysius K. Mok, and Louis E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *In Proceedings of the 11th Real-Time Systems Symposium*, pages 182–190, 1990. doi:10.1109/REAL.1990.128746.

11 Marko Bertogna, Michele Cirinei, and Giuseppe Lipari. New schedulability tests for real-time task sets scheduled by deadline monotonic on multiprocessors. In *Principles of Distributed Systems, 9th International Conference, OPODIS*, pages 306–321, 2005. doi:10.1007/11795490_24.

12 Enrico Bini and Giorgio C. Buttazzo. Schedulability analysis of periodic fixed priority systems. *IEEE Trans. Computers*, 53(11):1462–1473, 2004. doi:10.1109/TC.2004.103.

13 Enrico Bini and Giorgio C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005. doi:10.1007/s11241-005-0507-9.

14 Enrico Bini, Thi Huyen Chau Nguyen, Pascal Richard, and Sanjoy K. Baruah. A response-time bound in fixed-priority scheduling with arbitrary deadlines. *IEEE Trans. Computers*, 58(2):279–286, 2009. doi:10.1109/TC.2008.167.

15 Enrico Bini, Andrea Parri, and Giacomo Dossena. A quadratic-time response time upper bound with a tightness property. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 13–22, 2015. doi:10.1109/RTSS.2015.9.

16 Vincenzo Bonifaci, Ho-Leung Chan, Alberto Marchetti-Spaccamela, and Nicole Megow. Algorithms and complexity for periodic real-time scheduling. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1350–1359, 2010. doi:10.1137/1.9781611973075.109.

17 Jian-Jia Chen. Erratum: Global deadline-monotonic scheduling of arbitrary-deadline sporadic task systems, 2017. `http://ls12-www.cs.tu-dortmund.de/daes/media/documents/publications/downloads/2016-chen-erratum-globalDM.pdf`.

18 Jian-Jia Chen, Wen-Hung Huang, and Cong Liu. k2q: A quadratic-form response time and schedulability analysis framework for utilization-based analysis. In *Real-Time Systems Symposium, RTSS*, pages 351–362, 2016. `doi:10.1109/RTSS.2016.041`.

19 Jian-Jia Chen, Georg von der Brüggen, Wen-Hung Huang, and Robert I. Davis. On the pitfalls of resource augmentation factors and utilization bounds in real-time scheduling. In *Euromicro Conference on Real-Time Systems, ECRTS*, pages 9:1–9:25, 2017. `doi:10.4230/LIPIcs.ECRTS.2017.9`.

20 Jian-Jia Chen, Georg von der Brüggen, and Niklas Ueter. Evaluation results: Push forward: Global fixed-priority scheduling of arbitrary-deadline sporadic task systems. URL: `http://ls12-www.cs.tu-dortmund.de/daes/media/documents/publications/downloads/eval_push_forward.zip`.

21 Jian-Jia Chen, Georg von der Brüggen, and Niklas Ueter. Push forward: Global fixed-priority scheduling of arbitrary-deadline sporadic task systems, 2017. `http://ls12-www.cs.tu-dortmund.de/daes/media/documents/publications/downloads/2018-chen-ecrts-push-forward-full.pdf`.

22 Robert I. Davis and Alan Burns. Response time upper bounds for fixed priority real-time systems. In *Real-Time Systems Symposium (RTSS)*, pages 407–418, Nov 2008. `doi:10.1109/RTSS.2008.18`.

23 Robert I. Davis and Alan Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv.*, 43(4):35, 2011. `doi:10.1145/1978802.1978814`.

24 Friedrich Eisenbrand and Thomas Rothvoß. Static-priority real-time scheduling: Response time computation is np-hard. In *Proceedings of the 29th IEEE Real-Time Systems Symposium, RTSS*, pages 397–406, 2008. `doi:10.1109/RTSS.2008.25`.

25 Friedrich Eisenbrand and Thomas Rothvoß. EDF-schedulability of synchronous periodic task systems is coNP-hard. In *ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1029–1034, 2010. `doi:10.1137/1.9781611973075.83`.

26 Pontus Ekberg and Wang Yi. Uniprocessor feasibility of sporadic tasks remains conp-complete under bounded utilization. In *2015 IEEE Real-Time Systems Symposium, RTSS*, pages 87–95, 2015. `doi:10.1109/RTSS.2015.16`.

27 Pontus Ekberg and Wang Yi. Uniprocessor feasibility of sporadic tasks with constrained deadlines is strongly coNP-Complete. In *27th Euromicro Conference on Real-Time Systems, ECRTS*, pages 281–286, 2015. `doi:10.1109/ECRTS.2015.32`.

28 Paul Emberson, Roger Stafford, and Robert I Davis. Techniques for the synthesis of multiprocessor tasksets. In *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010)*, pages 6–11, 2010.

29 Gilles Geeraerts, Joël Goossens, and Markus Lindström. Multiprocessor schedulability of arbitrary-deadline sporadic tasks: complexity and antichain algorithm. *Real-time systems*, 49(2):171–218, 2013. `doi:10.1007/s11241-012-9172-y`.

30 Nan Guan, Martin Stigge, Wang Yi, and Ge Yu. New response time bounds for fixed priority multiprocessor scheduling. In *IEEE Real-Time Systems Symposium*, pages 387–397, 2009. `doi:10.1109/RTSS.2009.11`.

31 Wen-Hung Huang and Jian-Jia Chen. Response time bounds for sporadic arbitrary-deadline tasks under global fixed-priority scheduling on multiprocessors. In *International Conference on Real Time Networks and Systems, RTNS*, pages 215–224, 2015. `doi:10.1145/2834848.2834849`.

32 John P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *proceedings Real-Time Systems Symposium (RTSS)*, pages 201–209, Dec 1990. doi:10.1109/REAL.1990.128748.

33 C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973. doi:10.1145/321738.321743.

34 Lars Lundberg. Analyzing fixed-priority global multiprocessor scheduling. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 145–153, 2002. doi:10.1109/RTTAS.2002.1137389.

35 Mikael Sjodin and Hans Hansson. Improved response-time analysis calculations. In *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*, pages 399–408, 1998. doi:10.1109/REAL.1998.739773.

36 Youcheng Sun and Giuseppe Lipari. A pre-order relation for exact schedulability test of sporadic tasks on multiprocessor global fixed-priority scheduling. *Real-Time Systems*, 52(3):323–355, 2016. doi:10.1007/s11241-015-9245-9.

37 Youcheng Sun, Giuseppe Lipari, Nan Guan, and Wang Yi. Improving the response time analysis of global fixed-priority multiprocessor scheduling. In *International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 1–9, 2014. doi:10.1109/RTCSA.2014.6910543.

8 Appendix: Additional Proofs

8.1 Proof of Theorem 5.4: Speedup lower bound of global DM for arbitrary-deadline task systems

We will specifically use the following task set $\mathbf{T}^{ad}$ with $N = 2M + 1$ tasks. Let ε be an arbitrarily small positive real number such that $1/\varepsilon$ is an integer. Let $\eta \ll \varepsilon$ be an arbitrarily small positive number, that is used to enforce the priority assignment under global DM:

- $C_i = \frac{\varepsilon}{3}$, $T_i = \varepsilon$, $D_i = 1$, for $i = 1, 2, \ldots, M$.
- $C_i = \frac{1}{3}$, $T_i = \infty$, $D_i = 1 + \eta$, for $i = M + 1, M + 2, \ldots, 2M$.
- $C_i = \frac{1+\varepsilon}{3}$, $T_i = \infty$, $D_i = 1 + 2\eta$, for $i = 2M + 1$

As the setting of $\eta \ll \varepsilon$ is just to enforce the indexing, *we will directly take $\eta \to 0$ here.*

▶ **Lemma 8.1.** *$\mathbf{T}^{ad}$ is not schedulable by global DM.*

Proof. This can be proved by showing that task τ_N misses its deadline in the following concrete arrival pattern: all tasks release their first jobs at time 0 and the subsequent jobs arrive as early as possible while respecting their minimum inter-arrival times. For this arrival pattern, the jobs of tasks $\tau_1, \tau_2, \ldots, \tau_M$ are executed from time $i\varepsilon$ to time $i\varepsilon + \frac{\varepsilon}{3}$ for $i = 0, 1, 2, \ldots, 1/\varepsilon$. Therefore, these M tasks are executed for in total 1/3 time units from time 0 to time 1. For tasks $\tau_{1+M}, \tau_{2+M}, \ldots, \tau_{2M}$, each of them is executed for 1/3 time units from time 0 to time 1 when the processors do not execute $\tau_1, \tau_2, \ldots, \tau_M$. Task τ_{2M+1} is executed alone without any overlap with the executions of the higher-priority tasks. Therefore task τ_{2M+1} misses its deadline since it needs $\frac{1+\varepsilon}{3}$ time units, but only $\frac{1}{3}$ time units are available before its deadline. ◀

▶ **Lemma 8.2.** *There exists a feasible schedule for task set $\mathbf{T}^{ad}$ at any speed no lower than $\frac{1+\varepsilon}{3} + \frac{1+\varepsilon}{3M}$.*

Proof. We apply multiprocessor semi-partitioned scheduling, in which tasks in $\{\tau_m, \tau_{m+M}\}$ are assigned to processor m for $m = 1, 2, \ldots, M$. In our designed semi-partitioned schedule, a job of task τ_{2M+1}, i.e., a part of τ_N, is executed partially on each of the M processors as follows: it runs on processor m for C_N/M amount of time, and then migrates to processor $m+1$ to continue its execution, for $m = 1, 2, \ldots, M-1$. To ensure that the migration can be served immediately, τ_N is given the the highest-priority in this schedule. Therefore, a *subtask* of task τ_N on processor m, denoted as $\tau_{N,m}$, has a relative deadline C_N/M. As long as the speed of the processors is greater than or equal to $\frac{1+\varepsilon}{3}$, task τ_N can meet its deadline. Therefore, in our designed semi-partitioned schedule, each processor m has a task set $\mathbf{T}_m$ that consists of three tasks: τ_m and τ_{m+M} from $\mathbf{T}^{ad}$ and a subtask $\tau_{N,m}$ of task τ_N with execution time C_N/M. We assign the second priority to task τ_{m+M} and the lowest priority to task τ_m on processor m.

We utilize the worst-case response time analysis by Bini et al. [14]. They showed that if $1 - \sum_{\tau_i \in hp(\tau_k,m)} U_i \leq 1$, then the worst-case response time of a task τ_k in a task set $\mathbf{T}_m$ under fixed-priority scheduling on a processor is at most

$$\frac{C_k + \sum_{\tau_i \in hp(\tau_k,m)} C_i - \sum_{\tau_i \in hp(\tau_k,m)} U_i C_i}{1 - \sum_{\tau_i \in hp(\tau_k,m)} U_i}, \tag{38}$$

where $hp(\tau_k, m)$ is the set of the tasks in $\mathbf{T}_m$ that have a higher priorities than task τ_k. Note that the precondition $1 - \sum_{\tau_i \in hp(\tau_k,m)} U_i \leq 1$ for the test in Eq. (38) to be applicable always holds at any arbitrarily speed since we assign τ_m as the lower-priority task on processor m and $U_{m+M} \to 0$, and $U_{N,m} = C_{N,m}/T_N \to 0$.

By Eq. (38), if the speed of processor m is greater than or equal to $\frac{C_N}{M} + C_{m+M} = \frac{1+\varepsilon}{3M} + \frac{1}{3}$, task τ_{m+M} can still meet its deadline in this schedule. By Eq. (38), task τ_m can meet its deadline at speed s in this schedule if

$$1 \geq \frac{C_m/s + \sum_{\tau_i \in hp(\tau_k,m)} C_i/s - \sum_{\tau_i \in hp(\tau_k,m)} \frac{U_i}{s}\frac{C_i}{s}}{1 - \sum_{\tau_i \in hp(\tau_k,m)} U_i/s} = \frac{\varepsilon}{3s} + \frac{1}{3s} + \frac{1+\varepsilon}{3sM} \tag{39}$$

Therefore, as long as $s \geq \frac{1+\varepsilon}{3} + \frac{1+\varepsilon}{3M}$, task τ_m meets its deadline under our designed schedule. ◀