Laxity dynamics and LLF schedulability analysis on multiprocessor platforms

Jinkyu Lee · Arvind Easwaran · Insik Shin

Abstract LLF (Least Laxity First) scheduling, which assigns a higher priority to a task with a smaller laxity, has been known as an optimal preemptive scheduling algorithm on a single processor platform. However, little work has been made to illuminate its characteristics upon multiprocessor platforms. In this paper, we identify the dynamics of laxity from the system's viewpoint and translate the dynamics into LLF multiprocessor schedulability analysis. More specifically, we first character-ize laxity properties under LLF scheduling, focusing on laxity dynamics associated with a deadline miss. These laxity dynamics describe a lower bound, which leads to the deadline miss, on the number of tasks of certain laxity values at certain time instants. This lower bound is significant because it represents invariants for highly dynamic system parameters (laxity values). Since the laxity of a task is dependent of the amount of interference of higher-priority tasks, we can then derive a set of condi-tions to check whether a given task system can go into the laxity dynamics towards a deadline miss. This way, to the author's best knowledge, we propose the first LLF multiprocessor schedulability test based on its own laxity properties. We also de-

velop an improved schedulability test that exploits slack values. We mathematically prove that the proposed LLF tests dominate the state-of-the-art EDZL tests. We also present simulation results to evaluate schedulability performance of both the original and improved LLF tests in a quantitative manner.

Keywords Schedulability analysis · Laxity dynamics · LLF (Least Laxity First) · Multiprocessor platforms · Real-time scheduling

1 Introduction

Along with a recent trend towards multi-core architectures, real-time multi-core (multiprocessor) scheduling has been receiving a growing attention. Multiprocessor scheduling algorithms can be generally classified into two categories: partitioned and global. While partitioned algorithms restrict each task to run only on a designated processor, global algorithms allow each task to migrate from one processor to another. Many studies for both categories have been developed in order to exploit advantages of each category (see an excellent survey Davis and Burns 2011 for detailed comparison). In this paper, we restrict our attention to global scheduling.

Some multiprocessor studies in the past (e.g., Andersson et al. 2001; Cho et al. 2002; Srinivasan and Baruah 2002) have focused on adapting existing uniprocessor scheduling to multiprocessors, and some others have developed novel policies specific to multiprocessors (e.g., Baruah et al. 1996; Anderson and Srinivasan 2000; Cho et al. 2006; Andersson and Tovar 2006; Funaoka et al. 2008; Andersson and Bletsas 2008; Easwaran et al. 2009; Lee et al. 2010, 2011; Levin et al. 2010). In spite of some significant achievements of these studies, many important scheduling problems continue to pose challenges including the efficient scheduling of general task systems such as those in which task deadlines differ from their periods. Although optimal scheduling of general task systems has been shown to be impossible (Fisher et al. 2010), we cannot rule out the existence of better scheduling algorithms.

Assigning priority to tasks according to their deadlines (or "urgency") is a simple yet successful strategy for uniprocessor real-time scheduling. For example, EDF (Earliest Deadline First) (Liu and Layland 1973) and DM (Deadline Monotonic) (Leung and Whitehead 1982) are shown as optimal dynamic- and static-priority policies for preemptive uniprocessor scheduling. However, the sole focus on deadline can be no longer effective in multiprocessor scheduling. EDF and DM are thereby far from optimality on multiprocessors. As multiple tasks can run simultaneously on multiprocessors, we believe it becomes equally important to consider task "parallelism" when assigning priorities to tasks. Otherwise, a task may fail to meet a deadline even when a larger amount of processing capacity is allocated to the task than its processing demand, but it cannot fully utilize the allocations due to its own parallelism restriction (i.e., a task cannot run simultaneously on more than one processor).

Considering that a task with smaller time to deadline is more urgent and a task with larger execution time has more parallelism restriction (Lee et al. 2011), we believe that laxity is a good single parameter to capture both urgency and parallelism at the same time, where laxity of a task at any time is defined as remaining time to

deadline minus the amount of remaining execution. One successful example of laxity-based algorithms is EDZL (Earliest Deadline first until Zero Laxity) (Lee 1994) that is an extension of EDF with priority promotion in zero-laxity state. Since EDZL promotes the priority of any task that would otherwise inevitably miss a deadline, it dominates[1] EDF (Park et al. 2005). This property also holds for FPZL (Fixed Priority until Zero Laxity) (Davis and Burns 2011), an example of another laxity-based algorithm.

LLF (Least Laxity First) (Leung 1989) is an interesting scheduling algorithm that prioritizes tasks only according to their laxity; the smaller laxity, the higher priority. In contrast to EDF and EDZL, LLF naturally considers both urgency and parallelism through laxity all the time. Thereby, it is likely to perform better than EDF and EDZL on multiprocessor platforms. Figure 1 shows an example, where LLF generates a successful schedule for a task set while EDZL fails.[2] While Figure 1(a) shows that τ_6 misses a deadline at $t = 10$ under EDZL, the same task set is schedulable under LLF scheduling as shown in Subfigure (b). This is because LLF is able to pay attention to τ_3 even when its laxity is not yet zero. In addition to this example, we performed simulations for comparing the scheduling performance between LLF and other algorithms (EDZL (Lee 1994), Pfair (Baruah et al. 1996), and EDF (Liu and Layland 1973)) over constrained deadline task systems[3] (deadline no larger than task period). The system density (λ_{sys}) often serves as a measure of the overall processing demands of constrained deadline task systems, which will be formally defined in Sect. 2. When the system density does not exceed the number of processors (i.e., $\lambda_{sys} \leq m$), Pfair is known as optimal (Baruah et al. 1996) as shown in Fig. 2(a). However, Fig. 2(b) shows a different phenomenon that LLF significantly outperforms the others when $\lambda_{sys} > m$, while Pfair performs quite poorly in this case.

As such, the above examples and simulation results indicate that LLF is quite effective in multiprocessor scheduling, in particular, relatively more effective when $\lambda_{sys} > m$. However, it has so far received very little attention to analyze the schedulability of LLF on multiprocessors even though the optimality of LLF on a single processor has been analyzed (Dertouzos and Mok 1989; Leung 1989). Study of Phillips et al. (2002) shows that a task set feasible on m speed-1 processors is schedulable under LLF scheduling on both (a) m speed-$(2\text{-}1/m)$ processors and (b) $m + O(\log(\max_i C_i / \min_i C_i))$ speed-1 processors. To use this test as a schedulability test for LLF, however, a sufficient feasibility test is required. To our best knowledge, the only known technique to check feasibility is to use schedulability tests such as those described above. This means that any LLF test derived from Phillips et al. (2002) will be only as good as these previously known schedulability tests. Hence, in this paper, we aim to understand LLF multiprocessor scheduling and introduce the

[1] Scheduling algorithm (test) A dominates B if any task set deemed schedulable by B is also deemed schedulable by A, but the vice-versa is not true.

[2] While there are many task sets which are not schedulable by EDF and EDZL, but schedulable by LLF, there also exist a few task sets which have the opposite behavior (Bala Kalyanasundaram et al. 2000).

[3] Pfair is originally defined for implicit-deadline task systems such that each task's period (equal to deadline) is split into sub-deadlines with execution time of one unit. To adapt Pfair for constrained deadline task systems, we split each task's deadline into sub-deadlines.

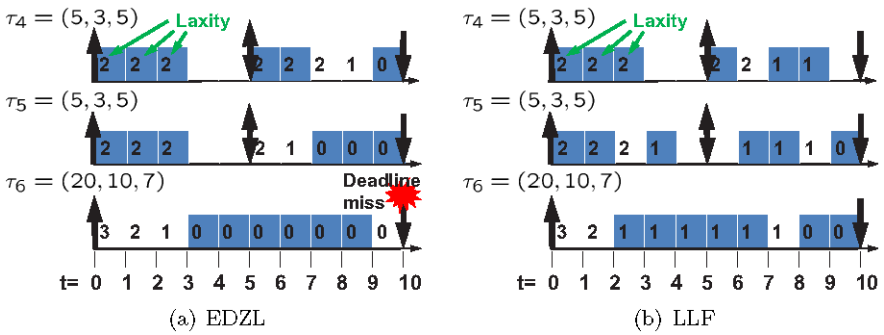


Fig. 1 EDZL and LLF schedules with a feasible task set on a two-processor platform: $\tau_4 = \tau_5 = (5, 3, 5)$, and $\tau_6 = (20, 7, 10)$, where (T_i, C_i, D_i) specifies the minimum separation, the worst-case execution time, and the relative deadline, respectively

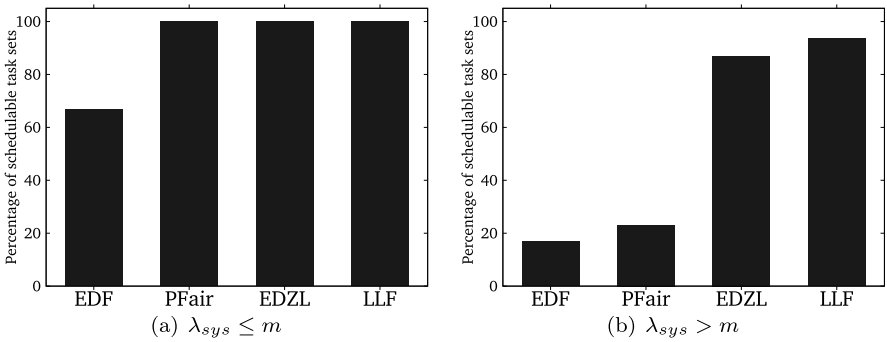


Fig. 2 This figure shows simulation results of various scheduling algorithms (LLF (Leung 1989), EDZL (Lee 1994), Pfair (Baruah et al. 1996), and EDF (Liu and Layland 1973)) over randomly generated 100,000 constrained deadline task sets on a four-processor platform (detailed set generation is described in Sect. 5.2). We perform simulation of each task set during the first 100, 000 time units for tractability, and thus the results show necessary schedulability

first LLF-specific schedulability test for multiprocessor platforms. For this purpose, we perform the following three steps.

1. We identify and define some properties associated with the LLF policy. We consider the scenario that a task misses its deadline at some time instant t_0, and compute the number of tasks that can have certain laxity values at certain time instants ahead of t_0. Lower bounds on these numbers that ensure the deadline miss are then used to define the properties. These properties are significant because they represent invariants (lower bounds) for highly dynamic system parameters (task laxity values).
2. We characterize these properties using conditions on the worst-case higher-priority interference, because previous studies suggest that it is feasible to derive multiprocessor schedulability tests using such conditions (e.g., Baker et al. 2008).

3. Using upper bounds on the higher-priority interference, derived in this paper and shown to be tighter than previously known bounds, we finally derive the LLF-specific schedulability test. By exploiting slack values for LLF, we also develop an improved LLF schedulability test.

To the best of our knowledge, these are the first LLF-specific schedulability tests for multiprocessors, and we show that they dominate the state-of-the-art EDZL schedulability tests (Baker et al. 2008). With these results, schedulability performance is also evaluated through simulations.

Extended version This paper is an extension of "LLF Schedulability Analysis on Multiprocessor Platforms" in the proceedings of the 31th IEEE Real-Time Systems Symposium (RTSS 2010) (Lee et al. 2010). In addition to re-organizing and re-phrasing the entire paper for the clarity of presentation, this paper includes the following new technical contributions, which have not presented in its preliminary conference version.

– We have developed an improved LLF test by the use of slack values with its time-complexity analysis. While existing approaches for other algorithms (e.g., EDF and EDZL) focus only on the interval between the release time and deadline of a task, we have developed a new way of calculating slack values for various interval lengths, which can be incorporated into necessary deadline miss conditions for LLF.
– We have evaluated the performance of our LLF schedulability tests with analytic and empirical comparison. In particular, we show average schedulability performance of our first and improved tests compared to corresponding EDF's and EDZL's tests under various parameters.
– We have discussed context switch overheads of LLF, and how to reduce such overheads.

Also, we have strengthened existing technical contents with error correction and details of omitted parts.

– We have extended the introduction with a discussion of why and how LLF is better than some others for multiprocessor scheduling.
– We have explicitly derived basic properties of LLF, which form the basis for deriving laxity dynamics.
– We have corrected Theorem 1 (which corresponds to Lemma 3 in Lee et al. 2010), one of the important principle in this paper. Then, we have presented the corrected and detailed proof of the theorem and its follow-up lemma (i.e., Lemma 4 which is also a core principle).
– We have presented a detailed procedure of reducing time-complexity of our schedulability tests.

Organization The rest of this paper is organized as follows. Section 2 describes our system model. Section 3 derives laxity dynamics when there is a deadline miss. Section 4 develops an LLF schedulability test and an improved test by capturing highly dynamic system parameters of laxity dynamics. Section 5 evaluates the performance

of the proposed LLF schedulability tests with analytic and empirical comparison. Section 6 discusses context switch overheads of LLF and how to reduce such overheads. Finally, Sect. 7 concludes this paper with discussion.

2 System model

Task model In this paper, we assume a sporadic task model (Mok 1983). In this model, we specify task $\tau_i \in \Gamma$ as (T_i, C_i, D_i), where T_i is the minimum separation, C_i is the worst-case execution time requirement, and D_i is the relative deadline. Task τ_i invokes a series of jobs, each separated from its predecessor by at least T_i time units. Further, we focus our attention to constrained ($C_i \leq D_i \leq T_i$) deadline task systems. We also assume that a single job of a task cannot be executed in parallel.

We use $D_i(t)$ and $C_i(t)$ to denote the remaining relative deadline and the remaining execution time, respectively, of a job of τ_i at time t. Note that since we focus on constrained deadline task systems, these quantities are well-defined. We express that a job of τ_i is *active* at t when $C_i(t)$ is non-zero. We use $L_i(t)$ to denote the laxity of a job of τ_i at t, and it is defined as $L_i(t) = D_i(t) - C_i(t)$. We denote the total number of tasks by n, and define the system utilization by $U_{sys} = \sum_{\tau_i \in \Gamma} C_i/T_i$ and the system density by $\lambda_{sys} = \sum_{\tau_i \in \Gamma} C_i/D_i$.

In this paper, we assume quantum-based time and without loss of generality let one time unit denote the quantum length. All task parameters are assumed to be specified as multiples of this quantum length. In constrained deadline task systems, at most one active job per task can exist in any time slot, and hence, for simplicity of presentation, we use the term "task" also to refer to "active job of a task" in the rest of the paper.

Multiprocessor platform We assume that the platform comprises m identical unit-capacity processors, and therefore restrict the system utilization U_{sys} to at most m. It has been previously shown that $U_{sys} \leq m$ is a necessary condition for feasibility of the task system considered here (Baker and Cirinei 2006). Like most existing studies in multiprocessor scheduling (for example, see Baruah et al. 1996), we assume that the system does not incur any penalty when a job is preempted or when a job is migrated from one processor to another.

3 Laxity dynamics

In this section, we present the dynamics of laxity under LLF scheduling. We first describe observations on the basic properties of LLF scheduling. Based on the properties, we then investigate how laxity values are changing over time when there is a deadline miss. Such laxity dynamics form the basis for new LLF schedulability tests, which are proposed in Sect. 4.

3.1 Basic properties of LLF

We first look at what happens to the laxity of task τ_i under LLF scheduling.

Observation 1 *The following properties hold under LLF scheduling.*

P1. A job can be blocked only by jobs with the same or smaller laxity values;
P2. A job of τ_i *satisfies* $L_i(t+1) = L_i(t)$, *if executed in* $[t, t+1)$;
P3. A job of τ_i *satisfies* $L_i(t+1) = L_i(t) - 1$, *if not executed in* $[t, t+1)$;
P4. A laxity value of a job is a non-increasing function of time; *and*
P5. A job with a negative laxity will miss its deadline eventually.

Since LLF assigns a higher priority to a job with a smaller laxity value, *P1* holds. Here we assume that a job can be blocked by jobs with even the *same* laxity values since we aim at designing schedulability tests applied to LLF with any tie-breaking rule. If a task is executed in the current time slot, both the remaining execution time and the remaining relative deadline of the task decrease by one. On the other hand, if not executed, only the remaining relative deadline of the task decreases. Therefore, a task's laxity value either stays the same or decreases by one in each time slot, resulting in *P2*, *P3*, and *P4*. Since a job with a negative laxity has larger remaining execution time than remaining relative deadline, *P5* holds. Using the properties *P1–P5*, we now derive laxity dynamics.

3.2 Laxity dynamics associated with deadline miss

For ease of presentation on laxity dynamics, we define a few notations. Let $S_\theta(t)$ denote a set of tasks whose active jobs have a laxity of θ at time instant t, and let $N_\theta(t)$ (≥ 0) denote the size of $S_\theta(t)$. Note that $S_{-1}(t)$ is defined to represent a set of tasks whose active jobs have a negative laxity. Suppose there is a job that misses its deadline, and let t_0 denote the first time instant when there is a job with a negative laxity. That is, t_0 is the first time instant such that $S_{-1}(t) \neq \emptyset$.

We now derive laxity dynamics of a task set scheduled by LLF before t_0. We first look at what would happen at $t_0 - 1$. By the definition of t_0, there is no job with a negative laxity at $t_0 - 1$. Then a task with a negative laxity at t_0 has a zero laxity at $t_0 - 1$, which implies the task is not scheduled in $[t_0 - 1, t_0)$. This means, at $t_0 - 1$, there are at least m other zero-laxity tasks, which implies there are at least $m + 1$ zero-laxity tasks in total. We record this in the following observation.

Observation 2 *The following condition holds under LLF*:

$$N_0(t_0 - 1) > m. \tag{1}$$

Note that the above observation holds generally for all zero-laxity based scheduling algorithms (that give the highest priority to tasks with zero laxity, e.g., EDZL (Lee 1994), FPZL (Davis and Burns 2011)). The next step is to consider what would happen at $t_0 - 2$ depending on what happens at $t_0 - 1$. We first consider a case where there is no job released or finished at $t_0 - 1$. By definition, there are $N_0(t_0 - 2)$ tasks with a zero laxity at $t_0 - 2$. We observe that $N_0(t_0 - 2) \leq m$ because otherwise $t_0 - 1$ should be the first instant when there is a task with a negative laxity, which is a contradiction to the definition of t_0. Thus, considering zero-laxity tasks have the highest priority under LLF, $N_0(t_0 - 2)$ zero-laxity tasks will be all scheduled in

$[t_0 - 2, t_0 - 1)$, and they will continue to have a zero laxity at $t_0 - 1$. In addition, some of the one-laxity tasks can be scheduled, and all the remaining tasks will not be scheduled. That is, $m - N_0(t_0 - 2)$ one-laxity tasks can be scheduled at $t_0 - 2$ together with $N_0(t_0 - 2)$ zero-laxity tasks. Hence, among $N_1(t_0 - 2)$ one-laxity tasks, $N_1(t_0 - 2) - (m - N_0(t_0 - 2))$ tasks will not be scheduled at $t_0 - 2$, and their laxity will become zero at $t_0 - 1$. Here we observe that $N_1(t_0 - 2) - (m - N_0(t_0 - 2)) > 0$. Suppose this is not true. Then, all tasks with a one or zero laxity at $t_0 - 2$ are scheduled at $[t_0 - 2, t_0 - 1)$, and this contradicts Observation 2. So the number of zero-laxity tasks at $t_0 - 1$ is given by

$$\begin{aligned} N_0(t_0 - 1) &= N_0(t_0 - 2) + N_1(t_0 - 2) - \big(m - N_0(t_0 - 2)\big) \\ &= 2 \cdot N_0(t_0 - 2) + N_1(t_0 - 2) - m. \end{aligned} \tag{2}$$

Extending Observation 2, we derive the following relationship between $N_0(t_0 - 2)$ and $N_1(t_0 - 2)$ as follows:

Observation 3 *If no job is released or finished at* $t_0 - 1$, *the following condition holds under LLF*:

$$\begin{aligned} & N_0(t_0 - 1) > m \\ & \iff 2 \cdot N_0(t_0 - 2) + N_1(t_0 - 2) - m > m \\ & \iff 2 \cdot N_0(t_0 - 2) + N_1(t_0 - 2) > 2 \cdot m. \end{aligned} \tag{3}$$

It is worth noting that the above observation is specific to LLF, and in particular, does not necessarily hold for other zero-laxity scheduling algorithms.

Now we consider the general case where there are jobs released and/or finished at $t_0 - 1$. We let $Z_\theta(t_0 - x)$ denote the number of tasks whose jobs are released at $t_0 - x + 1$ with a laxity of θ. We also let $W_\theta(t_0 - x)$ denote the number of tasks whose jobs are finished at $t_0 - x + 1$ and have a laxity of θ at $t_0 - x$. Like $N_\theta(t_0 - x)$, $Z_\theta(t_0 - x)$ and $W_\theta(t_0 - x)$ for $x = 1, 2, 3, \ldots, \infty$ are also all non-negative integer values.

We consider two cases: (A) $N_0(t_0 - 2) + N_1(t_0 - 2) > m$ and (B) $N_0(t_0 - 2) + N_1(t_0 - 2) \leq m$. If Case (A) holds, we can calculate $N_0(t_0 - 1)$ similarly as in Eq. (2):

$$N_0(t_0 - 1) = 2 \cdot N_0(t_0 - 2) + N_1(t_0 - 2) - m + Z_0(t_0 - 2) - W_0(t_0 - 2). \tag{4}$$

In this case, once we apply Eq. (4) to Inequality (1), we can get the following condition:

$$\begin{aligned} & N_0(t_0 - 1) > m \\ & \iff 2 \cdot N_0(t_0 - 2) + N_1(t_0 - 2) - m + Z_0(t_0 - 2) - W_0(t_0 - 2) > m \\ & \iff 2 \cdot N_0(t_0 - 2) + N_1(t_0 - 2) + Z_0(t_0 - 2) - W_0(t_0 - 2) > 2 \cdot m. \end{aligned} \tag{5}$$

On the other hand, if Case (B) holds (i.e., $N_0(t_0 - 2) + N_1(t_0 - 2) \leq m$), all tasks in $S_0(t_0 - 2)$ and $S_1(t_0 - 2)$ are scheduled in $[t_0 - 2, t_0 - 1)$ and continue to have the

same laxity values at $t_0 - 1$. This means no task in $S_1(t_0 - 2)$ will belong to $S_0(t_0 - 1)$, and thus $N_0(t_0 - 1)$ is given by

$$N_0(t_0 - 1) = N_0(t_0 - 2) + Z_0(t_0 - 2) - W_0(t_0 - 2). \tag{6}$$

In this case, we also apply Eq. (6) to Inequality (1), and then we can get the following condition:

$$\begin{aligned} & N_0(t_0 - 1) > m \\ & \quad \Longleftrightarrow N_0(t_0 - 2) + Z_0(t_0 - 2) - W_0(t_0 - 2) > m \\ & \quad \Longleftrightarrow 2 \cdot N_0(t_0 - 2) + 2 \cdot Z_0(t_0 - 2) - 2 \cdot W_0(t_0 - 2) > 2 \cdot m. \end{aligned} \tag{7}$$

To summarize, we present Inequalities (5) and (7) using a sufficient condition as follows.

Observation 4 *The following condition holds under LLF*:

$$2 \cdot N_0(t_0 - 2) + N_1(t_0 - 2) + (1 + \mathbf{1}_{t_0-2}) \cdot \big(Z_0(t_0 - 2) - W_0(t_0 - 2)\big) > 2 \cdot m, \tag{8}$$

$$\textit{where } \mathbf{1}_{t_0-x} = \begin{cases} 0, & \textit{if } \sum_{j=0}^{x-1} N_j(t_0 - x) > m \\ 1, & \textit{if } \sum_{j=0}^{x-1} N_j(t_0 - x) \le m. \end{cases} \tag{9}$$

Now we wish to generalize the above observation for all time instants before t_0, and present the following theorem.

Theorem 1 *Each of the following condition holds under LLF for* $x = 1, 2, 3, \ldots, \infty$:

$$\begin{aligned} & \sum_{j=0}^{x-1} (x - j) \cdot N_j(t_0 - x) \\ & + \sum_{k=2}^{x} \left(\prod_{p=k}^{x} \left(1 + \frac{\mathbf{1}_{t_0-p}}{p - 1} \right) \cdot \sum_{j=0}^{k-2} (k - j - 1) \cdot \big(Z_j(t_0 - k) - W_j(t_0 - k)\big) \right) > x \cdot m. \end{aligned} \tag{10}$$

Proof The basic idea of the proof is to use mathematical induction, and we consider two cases for each inductive step at $t_0 - x$: $\mathbf{1}_{t_0-x} = 1$ and $\mathbf{1}_{t_0-x} = 0$. A detailed proof is given in Appendix A. □

Based on these laxity dynamics, we will derive schedulability tests for LLF in the next section.

4 Schedulability test for LLF

In the previous section, we derived necessary conditions for a task to have a negative laxity at t_0 under LLF, in terms of the number of tasks with certain laxity values prior to t_0 (i.e., $N_\theta(t_0 - x)$) and the number of tasks released and finished prior to t_0 (i.e., $Z_\theta(t_0 - x)$ and $W_\theta(t_0 - x)$, respectively). However, we cannot calculate the highly dynamic system parameters $N_\theta(t_0 - x)$, $Z_\theta(t_0 - x)$ and $W_\theta(t_0 - x)$ without tracking all sporadic jobs' release times. In this section, we propose schedulability tests for LLF, which do not require such knowledge of release times. To do this, we first investigate whether a task can have a certain laxity or not. Based on this laxity qualification of each task, we investigate how to incorporate the necessary conditions into a LLF schedulability test by abstracting the parameters. Then, we propose an improved LLF schedulability test, which exploits slack values. Finally, we analyze time-complexity of our tests.

4.1 Laxity qualification

In this subsection, we derive sufficient conditions for a job of a given task to have a certain laxity value at a certain time unit ahead of its deadline. These conditions form the basis for abstracting $N_\theta(t_0 - x)$, $Z_\theta(t_0 - x)$ and $W_\theta(t_0 - x)$ for all possible values of θ and $t_0 - x$ in Sect. 4.2. To do this, we first derive the worst-case interference bound of jobs of task τ_i on a job of task τ_k in a time interval $[t_a, t_b)$, where t_a is the release time of τ_k's job, and t_b is some time instant no later than the deadline of the job (i.e., $t_b \leq t_a + D_k$).

In order to check whether a job of task τ_k can have a certain laxity at a certain time instant, we use the concept of the worst-case interference of higher-priority executions on the execution of a job of task τ_k between its release time and any arbitrary time instant before its deadline. Following the notations similar to existing studies (Bertogna et al. 2005, 2009; Baker et al. 2008; Lee et al. 2010, 2011), we denote the total interference of jobs of task τ_i on a job of task τ_k in an interval $[t_1, t_2)$ as $\overline{I}_{k \leftarrow i}(t_1, t_2)$. It represents the cumulative length of all intervals within $[t_1, t_2)$ in which a job of task τ_k is ready to execute and any job of task τ_i is executing while the job of task τ_k is not. The worst-case interference of jobs of task τ_i on a job of task τ_k in any interval of length l is then defined as

$$I_{k \leftarrow i}(l) = \max_t \overline{I}_{k \leftarrow i}(t, t + l), \tag{11}$$

and the overall worst-case higher priority interference on τ_k is defined as

$$\sum_{\tau_i \in \Gamma - \{\tau_k\}} I_{k \leftarrow i}(l). \tag{12}$$

Although the previously known interference bound for any work-conserving algorithm (Eq. (6) in Bertogna et al. 2009 and Eq. (4) in Lee et al. 2010) can also be used for LLF, we present a tighter LLF-specific interference bound. To derive such interference bound, we first introduce following lemmas.

Lemma 1 *Suppose task τ_i and task τ_k have the same laxity at t (i.e., $L_k(t) = L_i(t)$), and each task has a positive remaining execution time at t' ($> t$). Then, $-1 \le L_k(t') - L_k(t) \le 1$.*

Proof Suppose that $L_k(t') + 2 \le L_i(t')$. Since $L_k(t) = L_i(t)$, there exists $t \le t'' \le t'$ such that $L_k(t'') + 1 = L_i(t'')$. Since LLF gives a higher priority to a task with the same or smaller laxity, task τ_k cannot have higher priority than task τ_i as long as it has a larger laxity than τ_k. Therefore, $L_k(t') + 2 \le L_i(t')$ contradicts the prioritizing policy of LLF, and this proves the lemma. □

Lemma 2 *Suppose task τ_k has a laxity of θ or less at t_b, and the release time and the deadline of a carry-out job*[4] *of τ_i are $t_b + \alpha - D_i$ and $t_b + \alpha$, respectively ($\alpha \ge 0$). Then, the job of τ_i interferes with τ_k during at most $C_i - \max(0, \alpha - \min(\theta + 1, D_i - C_i))$.*

Proof We consider two cases: (A) there exists $t_b + \alpha - D_i \le t \le t_b$ such that $L_i(t) = L_k(t)$; and (B) there does not exist such t.

(Case A) By Lemma 1, $L_i(t_b)$ can be $\theta - 1$, θ, or $\theta + 1$. At t_b, $D_i(t_b) = \alpha$, and then, $C_i(t_b)$ can be $\alpha - \theta + 1$, $\alpha - \theta$, or $\alpha - \theta - 1$. This means that the amount of execution of τ_i done in $[t_b + \alpha - D_i, t_b)$ can be $C_i - \alpha + \theta - 1$, $C_i - \alpha + \theta$, or $C_i - \alpha + \theta + 1$, which are upper-bounded by $C_i - \max(0, \alpha - \min(\theta + 1, D_i - C_i))$. Therefore, the job of τ_i interfere with τ_k during at most $C_i - \max(0, \alpha - \min(\theta + 1, D_i - C_i))$.

(Case B-1) $L_i(t) > L_k(t)$ for all $t_b + \alpha - D_i \le t \le t_b$. Since τ_k's priority is always higher than τ_i's one, the amount of interference of τ_i on τ_k in $[t_b + \alpha - D_i, t_b)$ is zero.

(Case B-2) $L_i(t) < L_k(t)$ for all $t_b + \alpha - D_i \le t \le t_b$. Then, $L_i(t_b)$ is at most $\theta - 1$, which implies $C_i(t_b)$ is at least $\alpha - \theta + 1$, and then the amount of execution of τ_i done in $[t_b + \alpha - D_i, t_b)$ is at most $C_i - \alpha + \theta - 1$, which is upper-bounded by $C_i - \max(0, \alpha - \min(\theta + 1, D_i - C_i))$.

By Case (A), (B-1) and (B-2), the job of τ_i interfere with τ_k during at most $C_i - \max(0, \alpha - \min(\theta + 1, D_i - C_i))$. □

Thus, if we align the carry-out job of task τ_i such that $\alpha = \min(\theta + 1, D_i - C_i)$, the carry-out job of task τ_i can interfere with task τ_k during C_i time units. Figure 3 depicts such release patterns that result in the worse-case interference of task τ_i on task τ_k in $[t_a, t_b)$. That is, in Fig. 3, a job of τ_k is released at t_a and has a laxity of θ at t_b, and the carry-out job of τ_i in $[t_a, t_b)$ has its deadline at $t_b + \min(\theta + 1, D_i - C_i)$. We formally express the pattern as a function of l and θ, where l is the interval length and θ is a laxity value, which task τ_k has at the end of the interval.

$$I_{k \leftarrow i}(l, \theta) = \left\lfloor \frac{l'}{T_i} \right\rfloor C_i + \min\left(C_i, l' - \left\lfloor \frac{l'}{T_i} \right\rfloor T_i, l \right), \tag{13}$$

where $l' \stackrel{def.}{=} l + \min(\theta + 1, D_i - C_i)$.

[4] Here a carry-out job means it is released within the given interval, but its deadline is after the interval.

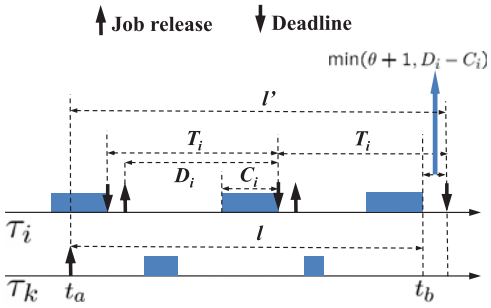


Fig. 3 Situation of the maximum interference of task τ_i on task τ_k in $[t_a, t_b)$ when $L_k(t_b) = \theta$

Then, using Eq. (13), we derive a sufficient condition for different possible laxity values of task τ_k and interval lengths as follows.

Lemma 3 *If task τ_k has a laxity of θ or less at y time units ahead of its deadline, then the following condition holds*:

$$\sum_{\tau_i \in \Gamma - \{\tau_k\}} I_{k \leftarrow i}(D_k - y, \theta) \geq m \cdot (D_k - C_k - \theta). \tag{14}$$

By Lemma 4 *in* Bertogna et al. (2005), *the above inequality can be further tightened as*:

$$\sum_{\tau_i \in \Gamma - \{\tau_k\}} \min\left(I_{k \leftarrow i}(D_k - y, \theta), D_k - C_k - \theta\right) \geq m \cdot (D_k - C_k - \theta). \tag{15}$$

Proof We prove this lemma by contraposition. That is, assuming $\sum_{\tau_i \in \Gamma - \{\tau_k\}} I_{k \leftarrow i} \times (D_k - y, \theta) < m \cdot (D_k - C_k - \theta)$, we prove that task τ_k must have a laxity larger than θ at y time units ahead of its deadline.

Note that τ_k cannot execute at some time instant only when m other tasks execute at that instant, and now the total interference on task τ_k is less than $m \cdot (D_k - C_k - \theta)$. Thus, during the interval $[t_a, t_a + D_k - y)$ (where t_a is the release time of τ_k's job), τ_k is prevented from executing for less than $m \cdot (D_k - C_k - \theta)/m$ time units due to interference. This means that τ_k executes for at least $D_k - y - (D_k - C_k - \theta - 1) = C_k + \theta + 1 - y$ time units in that interval. Then, the laxity of τ_k at $t_a + D_k - y$ can be computed as follows:

$$\begin{aligned} &D_k(t_a + D_k - y) = y \\ &C_k(t_a + D_k - y) \leq C_k - (C_k + \theta + 1 - y) = y - \theta - 1 \\ &L_k(t_a + D_k - y) \geq y - (y - \theta - 1) = \theta + 1. \end{aligned} \tag{16}$$

Therefore, task τ_k's laxity at y time units ahead of its deadline is strictly larger than θ. □

By Lemma 3, we know whether a task has a certain laxity at a certain time units ahead of its deadline. We now investigate how to incorporate a set of necessary conditions of Inequality (10) into LLF schedulability tests using Lemma 3.

For a given remaining relative deadline y ($\leq D_k$), we define the following indicator function $\delta_k^*(\theta, y)$ for task τ_k based on Lemma 3. This function indicates whether τ_k can reach a laxity of exactly θ at y time units ahead of its deadline.

$$\delta_k^*(\theta, y) = \begin{cases} 1, & \text{if this is the smallest } \theta \text{ for which Inequality (15) is true,} \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

Also, we define $\delta_k^*(\theta, y)$ for $y > D_k$ as follows:

$$\delta_k^*(\theta, y) = \begin{cases} 1, & \text{if } \theta = D_k - C_k, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Incorporating $\delta_k^*(\theta, y)$ into Theorem 1 we get the following lemma.

Lemma 4 *The following conditions hold under LLF for* $x = 1, 2, 3, \ldots, \infty$:

$$\sum_{j=0}^{x-1} (x - j) \sum_{\tau_k \in \Gamma} \delta_k^*(j, x) > x \cdot m. \tag{19}$$

Proof The basic idea of the proof is to show that the LHS of Inequality (19) is equal to or larger than the LHS of Inequality (10). Then Inequality (19) is a necessary condition of Inequality (10), and therefore this theorem holds. To do this, we investigate how much each task contributes to the LHS of Inequality (19) and that of Inequality (10). A detailed proof is given in Appendix B. □

Intuitively, the above lemma gives necessary conditions at $t_0 - x$ for a job to miss its deadline at t_0. For example, Eq. (19) for $x = 1$ means there should be more than m jobs which can have a zero laxity at one (i.e., $x = 1$) time unit ahead of its deadline. Similarly, Eq. (19) for $x = 2$ means it should hold that $2a + b > 2m$, where a (b) is the number of jobs which can have a zero (one) laxity at two (i.e., $x = 2$) time unit ahead of its deadline. Here note that Inequality (19) in Lemma 4, unlike Inequality (10) in Theorem 1, only depends on the task parameters and nothing else; in particular it is independent of time instant t_0.

Recall that the necessary conditions for a deadline miss in Lemma 4 are derived using constraints on the number of tasks with a certain laxity value at time instants $t_0 - 1$, $t_0 - 2$, …, where t_0 denotes the time instant when there exists at least one task with a negative laxity. Therefore the conditions in Lemma 4 can be further augmented with one more necessary condition characterizing one negative-laxity task at t_0 (i.e., $N_{-1}(t_0) \geq 1$). This condition can be also obtained by Lemma 3 with $y = 0$ and $\theta = -1$. Thus, incorporating this condition into Lemma 4, we formally express our LLF schedulability test as follow.

Theorem 2 (LLF schedulability test) *A task set is schedulable by LLF if at least one of Inequality* (19) *for* $x = 1, 2, 3, \ldots, D_{max}$, *and Condition* (20) *is not true, where* $D_{max} = \max_{\tau_i \in \Gamma}\{D_i\}$.

$$\textit{At least one task satisfies Inequality (15) with } y = 0 \textit{ and } \theta = -1. \tag{20}$$

Proof This theorem holds from Lemma 4. The difference between Lemma 4 and this theorem is the range of x. Since satisfying Inequality (19) in a limited range of x is a necessary condition for satisfying it in a more general range of x, correctness of this theorem holds trivially. Nevertheless, we now show that there is no need to investigate Inequality (19) for $x > D_{max}$. That is, assuming (19) holds for all $x \leq D_{max}$, we show that Inequality (19) holds for all $x > D_{max}$ by mathematical induction.

(The basis) Inequality (19) holds for all $x \leq D_{max}$.

(The inductive step) We will prove that if Inequality (19) for x holds, then Inequality (19) for $x + 1$ also holds when $x \geq D_{max}$. Since $x \geq D_k$ for all τ_k, only $\delta_k^*(\theta = D_k - C_k, y)$ terms are equal to 1 for both $y = x$ and $y = x + 1$. Thus, the LHS of Inequality (19) for $x + 1$ is increased by n (the number of tasks) compared to that of Inequality (19) for x, while the RHS of Inequality (19) for $x + 1$ is increased by m (the number of processors). It holds that $n > m$ to meet Inequality (19) for $x = 1$ is true, so Inequality (19) for $x + 1$ is also true.

We conclude that we do not need to investigate Inequality (19) for $x > D_{max}$. □

4.3 Improved LLF schedulability test

Using Lemma 3, we check whether the amount of interference on a task is large enough to have a certain laxity at a certain time instant before its deadline. However, the interference calculation function (Eq. (13)) contains pessimism in computing the worst-case interference. In particular, Eq. (13) does not consider the fact that a task can finish its execution earlier than its deadline. Once we identify such an early completion of a task's execution, we can reduce interference of a task using slack values, where the slack S_k of task τ_k is defined as a length of the minimum time interval between finishing time and deadline of a job of task τ_k. The idea of exploiting slack values is similar to those in Baker et al. (2008), Bertogna et al. (2009), but we extend previous techniques (Baker et al. 2008; Bertogna et al. 2009) to compute a lower bound on slack values. Those previous techniques consider only a single interval length (equal to the relative deadline of a task), while our technique explores various intervals of different lengths to obtain a tighter bound. From now on, we derive an improved LLF schedulability test by taking advantage of slack values. To do this, we first introduce how to calculate slack values for each task in the following lemma.

Lemma 5 *The slack of task* τ_k *is given by*

$$S_k = D_k - C_k - \theta - \left\lfloor \frac{\sum_{\tau_i \in \Gamma - \{\tau_k\}} \min(I_{k \leftarrow i}(D_k - y, \theta), D_k - C_k - \theta)}{m} \right\rfloor,$$

$$\textit{if } S_k \geq 1 \textit{ and } S_k \geq y - \theta. \tag{21}$$

Proof We re-arrange Eq. (21) as follows:

$$\left\lfloor \frac{\sum_{\tau_i \in \Gamma - \{\tau_k\}} \min(I_{k \leftarrow i}(D_k - y, \theta), D_k - C_k - \theta)}{m} \right\rfloor \cdot m = m \cdot (D_k - C_k - \theta - S_k)$$

$$\Longrightarrow \sum_{\tau_i \in \Gamma - \{\tau_k\}} \min\left(I_{k \leftarrow i}(D_k - y, \theta), D_k - C_k - \theta\right)$$

$$- m < m \cdot (D_k - C_k - \theta - S_k)$$

$$\Longleftrightarrow \sum_{\tau_i \in \Gamma - \{\tau_k\}} \min\left(I_{k \leftarrow i}(D_k - y, \theta), D_k - C_k - \theta\right)$$

$$< m \cdot (D_k - C_k - \theta - S_k + 1)$$

$$\Longrightarrow \sum_{\tau_i \in \Gamma - \{\tau_k\}} \min\left(I_{k \leftarrow i}(D_k - y, \theta), D_k - C_k - \theta - S_k + 1\right)$$

$$< m \cdot (D_k - C_k - \theta - S_k + 1). \tag{22}$$

The contraposition of Lemma 3 implies that if Inequality (22) is true, then task τ_k cannot have a laxity of $\theta + S_k - 1$ or less at y time units ahead of its deadline (denoted by t_1). Therefore, task τ_k can have at least $\theta + S_k$ laxity at t_1. At t_1, we obtain $C_k(t_1) \leq y - \theta - S_k$, by $D_k(t_1) = y$, and $L_k(t_1) \geq \theta + S_k$. Since we assume $S_k \geq y - \theta$ in this Lemma, it holds that $C_k(t_1) \leq 0$, which means task τ_k has already finished its execution at t_1. We consider two cases: (A) $S_k \leq y$ and (B) $S_k > y$. We let t_2 denote S_k time units ahead of τ_k's deadline. If Case (A) holds, $t_1 \leq t_2$, and then at t_2, τ_k does not have remaining execution time. If Case (B) holds (i.e., $t_2 < t_1$), and then $L_k(t_2) \geq L_k(t_1) \geq \theta + S_k$ by $P4$ in Observation 1. Here $D_k(t_2) = S_k$ by definition, which implies $C_k(t_2) \leq -\theta$. Therefore τ_k does not have remaining execution time at t_2. Therefore, task τ_k finishes its execution at S_k time units ahead of its deadline. □

We now exploit slack values for reducing interference of task τ_i on task τ_k. Considering that no execution of task τ_i is performed between S_i time units before its deadline, and its deadline, we substitute l'' in Fig. 4 for l' in Fig. 3. Here Fig. 4 depicts the release patterns for the worst-case interference of task τ_i on task τ_k in $[t_a, t_b)$. That is, in the figure, a job of τ_k is released at t_a and has a laxity of θ at t_b. In

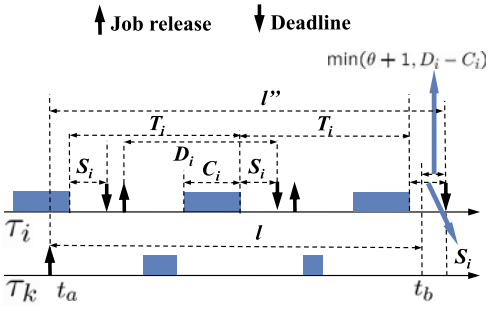


Fig. 4 Situation of the maximum interference of task τ_i on task τ_k in $[t_a, t_b)$ when $L_k(t_b) = \theta$ and there exists a positive slack value of task τ_i (S_i)

1. $S_i \leftarrow 0$, for all $\tau_i \in \Gamma$.
2. Do {
3. isHalt $\leftarrow$ true.
4. If Theorem 2 holds provided that Eq. (23) is applied to Inequality (15), return schedulable.
5. When Theorem 2 is tested, for each Inequality (15) with τ_i, θ, and y:
6. $S'_i \leftarrow$ the RHS of Eq. (21) if $S_i \geq 1$ and $S_i \geq y - \theta$.
7. If $S'_i > S_i$, then $S_i \leftarrow S'_i$ and isHalt $\leftarrow$ false.
8. } While isHalt = false.
9. Return unschedulable.

Fig. 5 Improved LLF schedulability test

addition, the carry-out job of τ_i in $[t_a, t_b)$ has its deadline at $t_b + \min(\theta + 1, D_i - C_i)$, and finishes its execution S_i ahead of its deadline. Then we derive a less pessimistic interference function than Eq. (13) by replacing l' with l'' as follows:

$$I'_{k \leftarrow i}(l, \theta) = \left\lfloor \frac{l''}{T_i} \right\rfloor C_i + \min\left(C_i, l'' - \left\lfloor \frac{l''}{T_i} \right\rfloor T_i, l\right), \tag{23}$$

where $l'' \stackrel{def.}{=} \max(0, l + \min(\theta + 1, D_i - C_i) - S_i)$.

Then, we reduce interference in an iterative manner, by increasing slack values. That is, at the first iteration, we calculate $\{S_i\}_{\tau_i \in \Gamma}$ based on Eq. (23) with $\{S_i = 0\}_{\tau_i \in \Gamma}$, and we continue updating $\{S_i\}_{\tau_i \in \Gamma}$ based on Eq. (23) with the previous $\{S_i\}_{\tau_i \in \Gamma}$ until there is no change between the previous and current $\{S_i\}_{\tau_i \in \Gamma}$. This iterative interference reduction is valid because $I'_{k \leftarrow i}(l, \theta)$ is a monotonically non-decreasing function of S_i for any given τ_i, τ_k, l and θ. Incorporating this technique to our LLF schedulability test, we develop an improved iterative LLF schedulability test. The detailed procedure of the improved test is described in Fig. 5.

4.4 Time-complexity

When we apply Theorem 2, calculation of the LHS of Inequality (15) for a given task τ_k, θ, and y has time-complexity $O(n)$. Given task τ_k, we need to calculate Inequality (15) for all pairs (θ, y) marked as $\surd$ in Table 1 in the worst-case, where the number of pairs is $O(C_k \cdot (D_k - C_k))$. Therefore, overall, the LLF schedulability test has time-complexity $O(n^2 \cdot \max_{\tau_i \in \Gamma} C_i \cdot (D_i - C_i))$. In fact, this complexity can be reduced to $O(n^2 \cdot \max_{\tau_i \in \Gamma} D_i)$, if we take advantage of properties associated with $\delta^*_k(\theta, y)$.

Before we describe how to reduce time-complexity, we define the following indicator function $\delta_k(\theta, y)$ of task τ_k, to indicate whether task τ_k may reach a laxity of θ at y ahead of its deadline:

$$\delta_k(\theta, y) = \begin{cases} 1, & \text{if Inequality (15) is true,} \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

We use the following relations; (A) if $\delta_k(\theta, y)$ is equal to 1, then $\delta_k(\theta + 1, y)$ and $\delta_k(\theta + 1, y + 1)$ are equal to 1; and (B) if $\delta_k(\theta, y)$ is equal to 0, then $\delta_k(\theta, y + 1)$ is

Table 1 A set of possible non-negative laxity values according to remaining relative deadline

Remaining time to deadline (y)	Laxity (θ)				
	0	1	...	$D_k - C_k - 1$	$D_k - C_k$
0					
1	$\surd$				
2	$\surd$	$\surd$			
...	$\surd$	$\surd$	...		
$D_k - C_k$	$\surd$	$\surd$	$\surd$	$\surd$	
$D_k - C_k + 1$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
...	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
C_k	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
$C_k + 1$		$\surd$	$\surd$	$\surd$	$\surd$
...			...	$\surd$	$\surd$
$D_k - 1$				$\surd$	$\surd$
D_k					$\surd$

equal to 0. These relations come from the definition of $\delta_k(\theta, y)$. For a task to have a laxity of θ at y ahead of its deadline, the task should be able to have a laxity of $\theta + 1$ at y and $y + 1$ ahead of its deadline. In turn, if a task cannot have a laxity of θ at y ahead of its deadline, it cannot have a laxity of θ at $y + 1$ ahead of its deadline, either. Using these relations, all $\delta_k(\theta, y)$ marked as $\surd$ in Table 1 can be determined by calculating $\delta_k(\theta, y)$ for at most D_k number of (θ, y) pairs as follows:

1. Start from $\theta = 0$ and $y = 1$.
2. Calculate $\delta_k(\theta, y)$.
3. If $\delta_k(\theta, y)$ is equal to 1, all $\delta_k(\theta + \alpha, y + \alpha)$ and $\delta_k(\theta + \alpha, y)$ are equal to 1 for $\alpha > 0$, and go to Step 5.
4. If $\delta_k(\theta, y)$ is equal to 0, all $\delta_k(\theta, y + \alpha)$ are equal to 0 for $\alpha > 0$, and go to Step 5.
5. If all $\delta_k(\theta, y)$ marked as $\surd$ in Table 1 are decided, then this procedure is finished. If not, we choose the smallest y, and then the smallest θ among all undecided $\delta_k(\theta, y)$ marked as $\surd$ in Table 1, and go to Step 2.

Once we know all available values of $\delta_k(\theta, y)$ in Table 1, we immediately know corresponding values of $\delta_k^*(\theta, y)$. With the above procedure, our LLF schedulability test in Theorem 2 has time-complexity $O(n^2 \cdot \max_{\tau_i \in \Gamma} D_i)$.

When we apply our improved LLF schedulability test in Fig. 5, each slack value S_i can be $0, 1, 2, \ldots$ or $(D_i - C_i)$. Then, in the worst case scenario, one iteration may update only one slack value by one, and thus the number of iterations is upper-bounded by $\sum_{\tau_i \in \Gamma}(D_i - C_i)$. Then, our improved schedulability test has time-complexity $O(n^2 \cdot \max_{\tau_i \in \Gamma} D_i \cdot \sum_{\tau_i \in \Gamma}(D_i - C_i))$.

5 Performance evaluation

This section evaluates the performance of the proposed LLF schedulability tests. First, we compare our tests with corresponding EDZL ones in an analytic manner.

Then, we presents simulation results to show average schedulability performance of our tests.

5.1 Analytic evaluation

It is not known whether there exists any dominance relationship between the LLF and EDZL scheduling algorithms themselves, but our LLF schedulability tests described in Theorem 2 and Fig. 5 dominate the corresponding EDZL schedulability tests in Baker et al. (2008). The following two lemmas record this.

Lemma 6 *The LLF schedulability test in Theorem 2 dominates the schedulability test of EDZL in Theorem 7 of* Baker et al. (2008).

Proof Conditions (6) and (7) in Theorem 7 of Baker et al. (2008) correspond to our two necessary conditions (19) for $x = 1$ and (20). The only difference is their interference functions. While conditions of EDZL (6) and (7) in Baker et al. (2008) use the same interference function as Eq. (13) with $l' = D_k$, our condition (20) uses Eq. (13) with $l' = D_k$ (i.e., $l = D_k$ and $\theta = -1$), and our another condition (19) for $x = 1$ uses Eq. (13) with $l' = D_k - 1 + \min(1, D_i - C_i)$ (i.e., $l = D_k - 1$ and $\theta = 0$). Therefore, the interference functions of our LLF test always give a value smaller than or equal to the interference function of the corresponding EDZL test, and then the LHS of conditions (19) for $x = 1$ and (20) is smaller than or equal to the LHS of conditions of EDZL (6) and (7). This proves the lemma. □

Lemma 7 *The improved LLF schedulability test in Fig. 5 dominates the improved (iterative) schedulability test of EDZL in Fig. 3 of* Baker et al. (2008).

Proof We can easily check that the slack values (S_i) of each task by Eq. (21) with both $(\theta, y) = (0, 1)$ and $(-1, 0)$ are equal to or larger than the slack value (S_i^{lb}) of each task by Eq. (12) of Baker et al. (2008). Then, while conditions of EDZL (6) and (7) in Baker et al. (2008) use the interference function of Eq. (13) with $l' = D_k - S_i$, condition (20) uses Eq. (13) with $l' = D_k - S_i$ and condition (19) for $x = 1$ uses Eq. (13) with $l' = D_k - 1 + \min(1, D_i - C_i) - S_i$. Therefore, the interference functions of our LLF test always gives a value smaller than or equal to the interference function of the corresponding EDZL test, and similar to Lemma 6, this proves the lemma. □

In addition to this dominance property, it is easily checked that our proposed tests in Theorem 2 and Fig. 5 are sustainable (Burns and Baruah 2008) in that a task set deemed schedulable by our tests is still deemed schedulable even if we increase D_k or T_k or decrease C_k.

5.2 Simulation settings

We generate task sets based on a technique proposed earlier (Baker 2005), which has been also used in many previous studies (e.g., Bertogna et al. 2009; Andersson et al.

2008). We have two input parameters: (A) the number of processors m (1, 2, 4, 8 or 16) and (B) individual task utilization (C_i/T_i) distribution (bimodal with parameter:[5] 0.1, 0.3, 0.5, 0.7, or 0.9, or exponential with parameter:[6] 0.1, 0.3, 0.5, 0.7, or 0.9). For each task, T_i is uniformly chosen in $[1, T_{max} = 1000]$, C_i is chosen based on the bimodal or exponential parameter, and D_i is uniformly chosen in $[C_i, T_i]$.

For each combination of (A) and (B), we repeat the following procedure (from Bertogna et al. 2009) and generate 100,000 task sets, thus resulting in 1,000,000 task sets for any given m.

1. Initially, we generate a set of $m + 1$ tasks.
2. In order to exclude unschedulable sets, we check whether the generated task set can pass a necessary feasibility condition (Baker and Cirinei 2006; Baruah et al. 2009).
3. If it fails to pass the feasibility test, we discard the generated task set and return to Step 1. Otherwise, we include this set for evaluation. Then, this set serves as a basis for the next new set; we create a new set by adding a new task into this old set and return to Step 2.

We evaluate the performance of six schedulability tests: (i) our proposed LLF test in Theorem 2, (ii) our improved (iterative) LLF test in Fig. 5, (iii) an EDZL test (Theorem 7 in Baker et al. 2008), (iv) an improved (iterative) EDZL test (Fig. 3 in Baker et al. 2008), (v) an EDF test (Theorem 7 in Bertogna et al. 2009), and (vi) an improved (iterative) EDF test (Sect. 6.2 in Bertogna et al. 2009). These tests are respectively annotated as 'LLF', 'LLF-I', 'EDZL', 'EDZL-I', 'EDF', and 'EDF-I' in the figures.

5.3 Simulation results

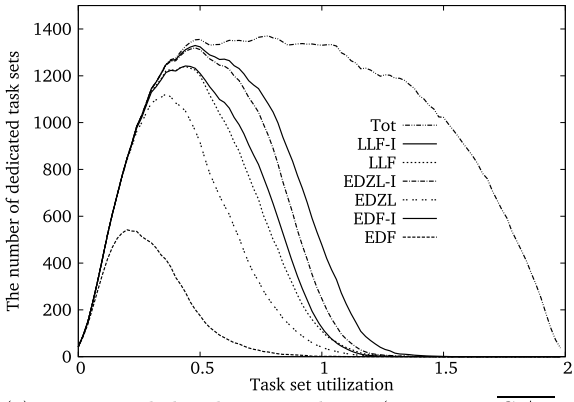
Figures 6 and 7 show schedulability test results of constrained deadline task sets for $m = 2$ and $m = 16$ over varying task utilization models. Since simulation results with other values of m (i.e., $m = 4, 8$) are similar to the figures, additional figures are not presented. Among the 10 task utilization models, we choose to show exponential distribution with 0.1, exponential distribution with 0.9, and bimodal distribution with 0.9, since they correspond to the cases, where the average task utilization ($\overline{C_i/T_i}$) is the smallest, medium, and the largest, respectively. Each figure comprises seven line-plots, each plot showing the number of task sets deemed schedulable by the corresponding tests over varying values of total task set utilization. In these figures, $\overline{n}$ denotes the average number of tasks per task set, and 'Tot' means the number of task sets with each total set utilization. The other six plots show the results of the six schedulability tests described in Sect. 5.2.

Figures 6 and 7 confirm Lemmas 6 and 7 in that they show LLF and LLF-I outperform corresponding EDZL tests (EDZL and EDZL-I) for all cases. We also observe

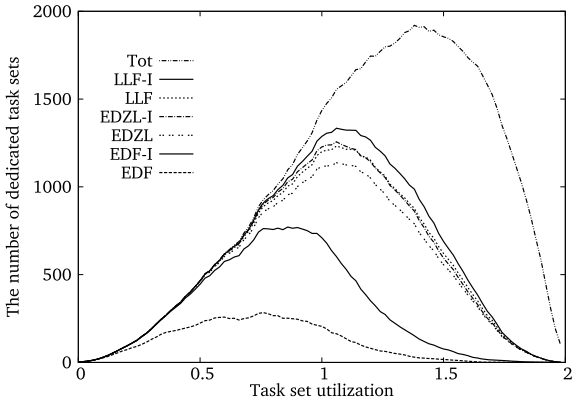
[5] For a given bimodal parameter p, a value for C_i/T_i is uniformly chosen in $[0, 0.5)$ with probability p, and in $[0.5, 1)$ with probability $1 - p$.

[6] For a given exponential parameter $1/\lambda$, a value for C_i/T_i is chosen according to the exponential distribution whose probability density function is $\lambda \cdot \exp(-\lambda \cdot x)$.

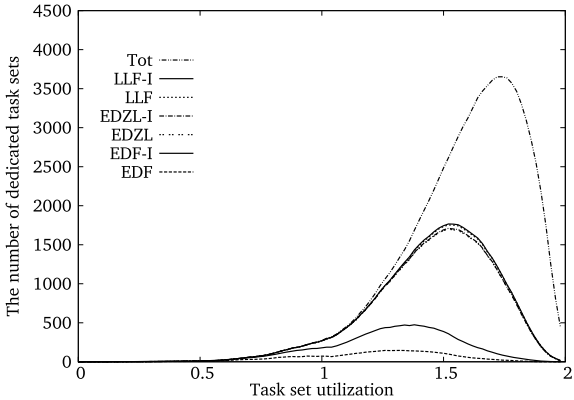
Fig. 6 Schedulability tests for $m = 2$



(a) Exponential distribution with 0.1 ($\overline{n} = 10.3$, $\overline{C_i/T_i} = 0.09$)



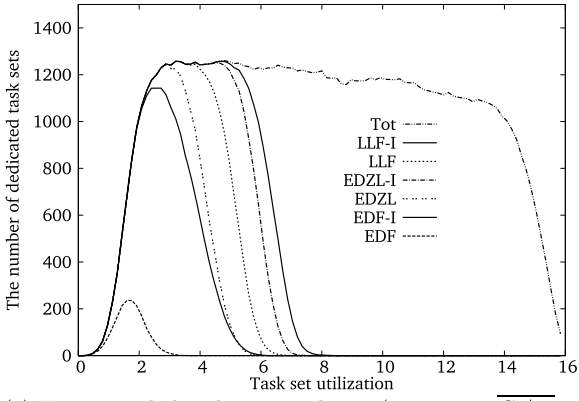
(b) Exponential distribution with 0.9 ($\overline{n} = 4.1$, $\overline{C_i/T_i} = 0.31$)



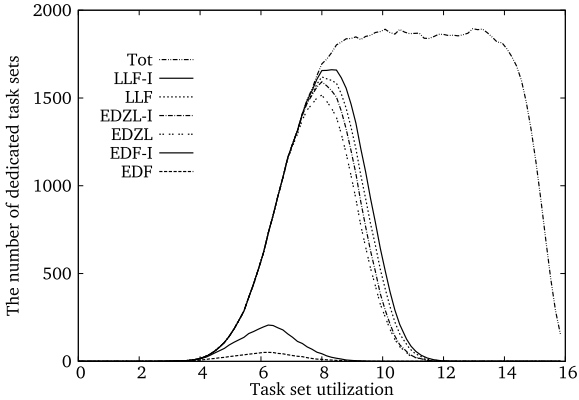
(c) Bimodal distribution with 0.9 ($\overline{n} = 3.1$, $\overline{C_i/T_i} = 0.52$)

that LLF-I always outperforms LLF, but the degree of improvement depends on average task utilization. That is, while there is a remarkable difference between LLF-I and LLF in Figs. 6(a) and 7(a), where average task utilization is the smallest, there

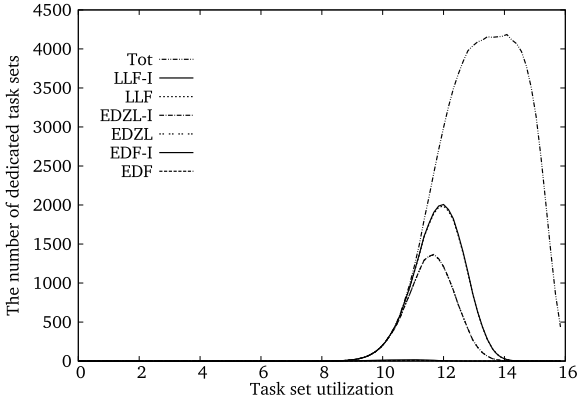
Fig. 7 Schedulability tests for $m = 16$



(a) Exponential distribution with 0.1 ($\overline{n} = 80.9$, $\overline{C_i/T_i} = 0.10$)



(b) Exponential distribution with 0.9 ($\overline{n} = 27.4$, $\overline{C_i/T_i} = 0.40$)



(c) Bimodal distribution with 0.9 ($\overline{n} = 19.4$, $\overline{C_i/T_i} = 0.69$)

is little difference in Figs. 6(c) and 7(c), where average task utilization is the largest. This is because a task with larger task utilization can have a smaller slack value than a task with smaller task utilization, and therefore if average task utilization is large, it is difficult to improve schedulability by slack values.

We now discuss two interesting observations from Figs. 6 and 7. First, in each of the six figures, we observe that the schedulability decreases as U_{sys} increases. This behavior is intuitive because $U_{sys} \leq m$ is a necessary feasibility condition for the task sets being considered here. Second, comparing the three figures for each value of m, we observe that the schedulability for a fixed value of U_{sys} decreases as the average task set utilization decreases (or the average number of tasks increases). For example, if we focus on the range $8 \leq U_{sys} \leq 12$ in Fig. 7, schedulability is different depending on average task set utilization; no task set is schedulable by LLF or LLF-I in Fig. 7(a), about 40 % of the task sets are schedulable in Fig. 7(b), and almost all the task sets are schedulable in Fig. 7(c). Similar behavior can also be observed for EDZL and EDZL-I. In Fig. 7(a), (b) and (c), the average number of tasks in each task set with $8 \leq U_{sys} \leq 12$ is respectively 97.2, 25.1 and 17.5, and the numbers indicate that schedulability at a fixed U_{sys} value decreases as the average number of tasks in each task set increases. This is because the pessimism in the interference bounds ($I'_{k \leftarrow i}(l, \theta)$ in Eq. (23)) is more pronounced with more tasks. As a result it is more likely that Inequality (15) will falsely identify a task to be capable of achieving a certain laxity value within a certain time ahead of its deadline, thus making our tests (as well as EDZL schedulability tests) more pessimistic. Note that this behavior has been also observed in the schedulability test for the fixed priority scheduling algorithm (Davis and Burns 2011).

With the scheduling performance, it is worth presenting the time-complexity of each schedulability test. As mentioned in Sect. 4.4, LLF and LLF-I have $O(n^2 \cdot \max_{\tau_i \in \Gamma} D_i)$ and $O(n^2 \cdot \max_{\tau_i \in \Gamma} D_i \cdot \sum_{\tau_i \in \Gamma}(D_i - C_i))$ time-complexity, respectively. On the other hands, tests for EDF and EDZL only need to investigate whether a task can have a negative laxity (or a zero laxity), and the time-complexity of them does not include the term of $\max_{\tau_i \in \Gamma} D_i$. Therefore EDF and EDZL, and EDF-I and EDZL-I have $O(n^2)$, and $O(n^2 \cdot \sum_{\tau_i \in \Gamma}(D_i - C_i))$ time-complexity, respectively (Bertogna et al. 2009; Baker et al. 2008).

6 Discussion

As shown in the previous section, LLF, changing the priorities of jobs dynamically, has a great potential to improve schedulability. However, such dynamic priority assignments can cause jobs to preempt each other frequently. For example, consider two jobs that have the same remaining execution time C and the same laxity L at some time instant on a single processor. Then, the LLF scheduler will execute those two jobs alternatively yielding as many preemptions as C per each job. This could negate the potential for schedulability improvement in some environments, where context switch and migration impose significant overheads.

For such environments, let us discuss a variant of LLF scheduling, called "Least Laxity-Group First" (LLGF), to reduce the number of preemptions. A laxity group

$\mathcal{L}$ can be defined as an integer value, and we assume there is a laxity-group mapping function $f(L)$ that translates some laxity L to some laxity group $\mathcal{L}$. For example, $f(L)$ can be defined as

$$\mathcal{L} = f(L) = \left\lceil \frac{L}{\alpha} \right\rceil,$$

where α represents the laxity group size. Then, LLGF schedules jobs according to their laxity groups, and ties can be broken arbitrarily. When ties are broken according to EDF, LLGF is a generalization of LLF and EDZL: if $\alpha = 1$, it is the same as LLF; and if $\alpha \geq \max_{\tau_i \in \mathcal{T}}(D_i - C_i)$, it is equivalent to EDZL. LLGF then incurs less preemptions than LLF does, since a job can preempt other jobs at most once per each laxity group value under LLGF but it can preempt other jobs every different laxity value under LLF. That is, a job of τ_i can preempt other jobs at most $\min(C_i, D_i - C_i + 1)$ times under LLF, but only at most $\min(C_i, \lceil \frac{D_i - C_i}{\alpha} \rceil + 1)$ times under LLGF. Therefore, we can reduce the number of preemptions by increasing α.

This raises many issues, including a new schedulability analysis for LLGF, exploring the impact of α on schedulability and preemptions, and finding a good value of α for some given context switch and migration overheads. Those issues are beyond the scope of this paper, while this paper focuses on the schedulability analysis of LLF. However, we believe the LLF analysis proposed in this paper will be used as a basis for the schedulability analysis for LLGF and further issues.

7 Conclusion

In this work, we have identified laxity dynamics of the LLF scheduling algorithm, and then derived the first LLF-specific schedulability tests for unit-capacity multiprocessor platforms. Dominance of these tests over previously known tests for EDZL has also been established, and its effectiveness has been demonstrated through simulations.

LLF has a great potential to improve schedulability at the risk of incurring more preemptions. We plan to develop variants of LLF scheduling, such as LLGF explained in Sect. 6, to balance the tradeoff between schedulability improvement and context switch/migration costs.

Acknowledgements This work was supported in part by Basic Research Laboratory Program (BRL, 2009-0086964), Basic Science Research Program (2010-0006650), P^3 DigiCar Research Center (NCRC, 2012-0000980), IT/SW Creative research program (NIPA-2010-C1810-1102-0003), SW Computing R&D Program of KEIT (2011-10041313), and Global Collaborative R&D program of KIAT (M002300089) funded by the Korea Government (MEST/MKE), and KAIST-Microsoft Research Collaboration Center.

Appendix A: Proof of Theorem 1

The basic idea of the proof is to use mathematical induction.

(Basis) Each Inequality (10) for $x = 1$ and $x = 2$ holds by Observations 2 and 4, respectively.

(Inductive step) We will prove the following statements: for all $x \geq 2$, if Inequality (10) for x holds, then Inequality (10) for $x+1$ also holds. We consider two cases depending on the value of $\mathbf{1}_{t_0-x-1}$.

(Inductive step: case A) Assume $\mathbf{1}_{t_0-x-1} = 1 \iff \sum_{j=0}^{x} N_j(t_0-x-1) \leq m$.

All tasks in $S_j(t_0-x-1)$ for $0 \leq j < x$ are serviced in $[t_0-x-1, t_0-x)$, and thus $L_k(t_0-x-1) = L_k(t_0-x)$ for all tasks $\tau_k \in S_j(t_0-x-1)$ where $0 \leq j < x$. So, the following condition holds:

$$N_j(t_0-x) = N_j(t_0-x-1) + Z_j(t_0-x-1) - W_j(t_0-x-1), \quad \forall 0 \leq j < x. \tag{25}$$

Applying Eq. (25) to Inequality (10), we get the following condition:

$$\begin{aligned}
&\sum_{j=0}^{x-1}(x-j)\cdot N_j(t_0-x-1) \\
&\quad + \sum_{j=0}^{x-1}(x-j)\cdot\big(Z_j(t_0-x-1) - W_j(t_0-x-1)\big) \\
&\quad + \sum_{k=2}^{x}\left(\prod_{p=k}^{x}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k) - W_j(t_0-k)\big)\right) > x\cdot m.
\end{aligned} \tag{26}$$

Multiplying the above condition by $\frac{x+1}{x}$, we get the following condition:

$$\begin{aligned}
&\sum_{j=0}^{x-1}\frac{x+1}{x}(x-j)\cdot N_j(t_0-x-1) \\
&\quad + \sum_{j=0}^{x-1}\frac{x+1}{x}(x-j)\cdot\big(Z_j(t_0-x-1) - W_j(t_0-x-1)\big) \\
&\quad + \sum_{k=2}^{x}\left(\frac{x+1}{x}\prod_{p=k}^{x}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k) - W_j(t_0-k)\big)\right) \\
&> (x+1)\cdot m.
\end{aligned} \tag{27}$$

We now look at the three terms of the LHS of Inequlaity (27). The first term can be upper-bounded as follows:

$$\sum_{j=0}^{x-1}\frac{x+1}{x}(x-j)\cdot N_j(t_0-x-1) \leq \sum_{j=0}^{x-1}(x+1-j)\cdot N_j(t_0-x-1) \tag{28}$$

This is because we observe the following condition for $x \geq 1$, $x \geq j \geq 0$:

$$\frac{x+1}{x}(x-j) = x+1-\frac{x+1}{x}j \leq x+1-j. \tag{29}$$

If $p = x+1$, then $\mathbf{1}_{t_0-p} = 1$ and $(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}) = \frac{x+1}{x}$. Therefore, the second term of the LHS of Inequlaity (27) can be rearranged if $k = x+1$ as follows:

$$\begin{aligned}
&\sum_{j=0}^{x-1} \frac{x+1}{x}(x-j)\cdot\big(Z_j(t_0-x-1) - W_j(t_0-x-1)\big) \\
&\quad= \sum_{j=0}^{k-2} \frac{x+1}{x}(k-j-1)\cdot\big(Z_j(t_0-k) - W_j(t_0-k)\big) \\
&\quad= \sum_{j=0}^{k-2}\left(\prod_{p=k}^{x+1}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\right)\cdot(k-j-1)\cdot\big(Z_j(t_0-k) - W_j(t_0-k)\big) \\
&\quad= \prod_{p=k}^{x+1}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k) - W_j(t_0-k)\big). \qquad (30)
\end{aligned}$$

And, the third term of the LHS of Inequality (27) can be also rearranged as follows:

$$\begin{aligned}
&\sum_{k=2}^{x}\left(\frac{x+1}{x}\prod_{p=k}^{x}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k) - W_j(t_0-k)\big)\right) \\
&\quad= \sum_{k=2}^{x}\Bigg(\prod_{p=x+1}^{x+1}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\prod_{p=k}^{x}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right) \\
&\qquad\times\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k) - W_j(t_0-k)\big)\Bigg) \\
&\quad= \sum_{k=2}^{x}\left(\prod_{p=k}^{x+1}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k) - W_j(t_0-k)\big)\right). \\
&\qquad\qquad (31)
\end{aligned}$$

Then, once we apply Eqs. (28), (30) and (31) to Inequality (27), then the LHS of Inequality (27) can be upper-bounded as follows:

$$\begin{aligned}
&\text{the LHS of Inequality (27)} \\
&\le \sum_{j=0}^{x-1}(x+1-j)\cdot N_j(t_0-x-1) \\
&\quad+\sum_{k=2}^{x+1}\left(\prod_{p=k}^{x+1}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k) - W_j(t_0-k)\big)\right). \\
&\qquad\qquad (32)
\end{aligned}$$

We can now check that Inequality (10) holds for $x+1$ by adding the non-negative term for $j = x$ to the first summation of the RHS of Inequality (32). Finally, we conclude that if Inequality (10) for x is true, then Inequality (10) for $x+1$ is also true when $\mathbf{1}_{t_0-x-1} = 1$.

(Inductive step: case B) Assume $\mathbf{1}_{t_0-x-1} = 0 \iff \sum_{j=0}^{x} N_j(t_0 - x - 1) > m$.

Since at most m tasks can be serviced in $[t_0-x-1, t_0-x)$, there exists a minimum number y ($\leq x$) such that at least one of the tasks in $S_y(t_0 - x - 1)$ is not serviced in $[t_0 - x - 1, t_0 - x)$. It holds that $y \geq 1$ because otherwise $N_0(t_0 - x - 1) > m$, which means $t_0 - x$ is the first instant when there is a task with a negative laxity. Thus, all tasks in $S_j(t_0 - x - 1)$ for $j = 0, \ldots, y-1$ are serviced while all tasks in $S_j(t_0 - x - 1)$ for $j = y+1, \ldots, x$ are not serviced. Among tasks in $S_y(t_0 - x - 1)$, $\sum_{j=0}^{y} N_j(t_0 - x - 1) - m$ tasks are not serviced and $m - \sum_{j=0}^{y-1} N_j(t_0 - x - 1)$ tasks are serviced. Considering that serviced tasks keep their laxity values and non-serviced ones reduce their laxity values by one, we establish the following relationship between $N_j(t_0 - x)$ and $N_j(t_0 - x - 1)$:

$$N_j(t_0 - x) = Q + Z_j(t_0 - x - 1) - W_j(t_0 - x - 1), \tag{33}$$

where

$$Q = \begin{cases} N_j(t_0 - x - 1), & 0 \leq j \leq y-2, \\ N_j(t_0 - x - 1) + (\sum_{k=0}^{y} N_k(t_0 - x - 1) - m), & j = y-1, \\ (m - \sum_{k=0}^{y-1} N_k(t_0 - x - 1)) + N_{j+1}(t_0 - x - 1) & j = y, \\ N_{j+1}(t_0 - x - 1), & y+1 \leq j \leq x-1. \end{cases}$$

The detailed calculation of Q is as follows. For $j = 0, \ldots, y-1$, all tasks in $S_j(t_0 - x - 1)$ keep their laxity values at $t_0 - x$, and therefore the first two cases include $N_j(t_0 - x - 1)$ terms. Among tasks in $S_y(t_0 - x - 1)$, $(\sum_{k=0}^{y} N_k(t_0 - x - 1) - m)$ tasks do not perform their executions in $[t_0 - x - 1, t_0 - x)$ and then each of their laxity values is $y - 1$ at $t_0 - x$. In turn, $(m - \sum_{k=0}^{y-1} N_k(t_0 - x - 1))$ tasks perform its execution in $[t_0 - x - 1, t_0 - x)$ and then each of their laxity values is y at $t_0 - x$. Therefore, the second and third cases have the corresponding terms. For $j = y, \ldots, x-1$, all tasks in $S_{j+1}(t_0 - x + 1)$ do not perform their executions in $[t_0 - x - 1, t_0 - x)$ and then each of their laxity values is j at $t_0 - x$. Thus, the third and fourth cases includes $N_{j+1}(t_0 - x - 1)$ terms.

By putting the above equation into Inequality (10), we derive the following condition:

$$\begin{aligned} &\sum_{j=0}^{y-2} (x - j) \cdot N_j(t_0 - x - 1) \\ &\quad + \big(x - (y-1)\big) \cdot \left(N_{y-1}(t_0 - x - 1) + \sum_{k=0}^{y} N_k(t_0 - x - 1) - m \right) \\ &\quad + (x - y) \cdot \left(m - \sum_{k=0}^{y-1} N_k(t_0 - x - 1) + N_{y+1}(t_0 - x - 1) \right) \end{aligned}$$

$$
\begin{aligned}
&+ \sum_{j=y+1}^{x-1} (x-j) \cdot N_{j+1}(t_0 - x - 1) \\
&+ \sum_{j=0}^{x-1} (x-j) \cdot \big(Z_j(t_0 - x - 1) - W_j(t_0 - x - 1)\big) \\
&+ \sum_{k=2}^{x} \left(\prod_{p=k}^{x} \left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot \sum_{j=0}^{k-2} (k-j-1) \cdot \big(Z_j(t_0-k) - W_j(t_0-k)\big) \right) \\
&> x \cdot m. \qquad (34)
\end{aligned}
$$

We re-arrange terms as follows:

$$
\begin{aligned}
&\sum_{j=0}^{y-1} (x-j) \cdot N_j(t_0 - x - 1) + \sum_{j=y}^{x-1} (x-j) \cdot N_{j+1}(t_0 - x - 1) \\
&\quad + \big(x - (y-1)\big) \cdot \sum_{j=0}^{y} N_j(t_0 - x - 1) - (x-y) \cdot \sum_{j=0}^{y-1} N_j(t_0 - x - 1) \\
&\quad - \big(x - (y-1)\big) \cdot m + (x-y) \cdot m \\
&\quad + \sum_{j=0}^{x-1} (x-j) \cdot \big(Z_j(t_0 - x - 1) - W_j(t_0 - x - 1)\big) \\
&\quad + \sum_{k=2}^{x} \left(\prod_{p=k}^{x} \left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot \sum_{j=0}^{k-2} (k-j-1) \cdot \big(Z_j(t_0-k) - W_j(t_0-k)\big) \right) \\
&> x \cdot m. \qquad (35)
\end{aligned}
$$

To re-arrange the first two lines of the above inequality, we use the following trivial equations.

- $\sum_{j=y}^{x-1} (x-j) \cdot N_{j+1}(t_0 - x - 1)$ is equal to $\sum_{j=y+1}^{x} (x-j+1) \cdot N_j(t_0 - x - 1)$.
- $(x - (y-1)) \cdot \sum_{j=0}^{y} N_j(t_0 - x - 1) - (x-y) \cdot \sum_{j=0}^{y-1} N_j(t_0 - x - 1)$ is equal to $(x - (y-1)) \cdot N_y(t_0 - x - 1) + \sum_{j=0}^{y-1} N_j(t_0 - x - 1)$

Using the above equations, we derive the following equation.

The sum of the first two lines of the Inequality (35)

$$
\begin{aligned}
&= \sum_{j=0}^{y-1} (x-j) \cdot N_j(t_0 - x - 1) + \sum_{j=y+1}^{x} (x-j+1) \cdot N_j(t_0 - x - 1) \\
&\quad + \big(x - (y-1)\big) \cdot N_y(t_0 - x - 1) + \sum_{j=0}^{y-1} N_j(t_0 - x - 1)
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{j=0}^{y-1}(x-j+1)\cdot N_j(t_0-x-1) + \sum_{j=y+1}^{x}(x-j+1)\cdot N_j(t_0-x-1) \\
&\quad + \big(x-(y-1)\big)\cdot N_y(t_0-x-1) \\
&= \sum_{j=0}^{x}(x-j+1)\cdot N_j(t_0-x-1). \qquad (36)
\end{aligned}
$$

We now re-arrange the fourth and fifth lines of the Inequality (35). Using $\mathbf{1}_{t_0-x-1}=0$, we obtain $\prod_{p=x+1}^{x+1}(1+\frac{\mathbf{1}_{t_0-p}}{p-1})=1$, and then the fourth line is equal to $\prod_{p=x+1}^{x+1}(1+\frac{\mathbf{1}_{t_0-p}}{p-1})\sum_{j=0}^{x-1}(x-j)\cdot(Z_j(t_0-x-1)-W_j(t_0-x-1))$. Once we substitute $k=x+1$, the fourth line is equal to $\prod_{p=x+1}^{k}(1+\frac{\mathbf{1}_{t_0-k}}{k-1})\sum_{j=0}^{k-2}(k-j-1)\cdot(Z_j(t_0-k)-W_j(t_0-k))$. Then,

The sum of the fourth and fifth lines of the Inequality (35)

$$
\begin{aligned}
&= \prod_{p=x+1}^{k}\left(1+\frac{\mathbf{1}_{t_0-k}}{k-1}\right)\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k)-W_j(t_0-k)\big) \\
&\quad + \sum_{k=2}^{x}\left(\prod_{p=k}^{x}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k)-W_j(t_0-k)\big)\right) \\
&= \sum_{k=2}^{x+1}\left(\prod_{p=k}^{x+1}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k)-W_j(t_0-k)\big)\right). \\
&\qquad (37)
\end{aligned}
$$

We add m for both the LHS and RHS of the Inequality (35) to eliminate the third line, and replace the first two lines, and the third and fourth lines by Eqs. (36) and (37), respectively. We finally obtain the following inequality.

Inequality (35)

$$
\begin{aligned}
&\iff \sum_{j=0}^{x}(x-j+1)\cdot N_j(t_0-x-1) \\
&\quad + \sum_{k=2}^{x+1}\left(\prod_{p=k}^{x+1}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k)-W_j(t_0-k)\big)\right) \\
&> (x+1)\cdot m. \qquad (38)
\end{aligned}
$$

The last condition is identical to Inequality (10) for $x+1$. Finally, we conclude that if Inequality (10) for x is true, then Inequality (10) for $x+1$ is also true when $\mathbf{1}_{t_0-x-1}=0$.

By (Inductive step: case A) and (Inductive step: case B), the inductive step is correct.

Appendix B: Proof of Lemma 4

We show that the LHS of Inequality (19) is equal to or larger than that of Inequality (10) for $x = 1, 2, 3, \ldots, \infty$. Then Inequality (19) is a necessary condition of Inequality (10), and then the lemma directly follows from Theorem 1, where Inequality (19) is as follows:

$$\sum_{j=0}^{x-1}(x-j)\sum_{\tau_k\in\Gamma}\delta_k^*(j,x) > x\cdot m,$$

and Inequality (10) is as follows:

$$\sum_{j=0}^{x-1}(x-j)\cdot N_j(t_0-x) + \sum_{k=2}^{x}\left(\prod_{p=k}^{x}\left(1+\frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot\sum_{j=0}^{k-2}(k-j-1)\cdot\big(Z_j(t_0-k)-W_j(t_0-k)\big)\right) > x\cdot m.$$

We investigate how much individual tasks contribute to the LHS of Inequality (10) and to that of Inequality (19) for given x. Then we prove that the contribution of task τ_k to the LHS of Inequality (19) is always equal to or larger than that to the LHS of Inequality (10). We let (A_x) and (B_x) denote the LHS of Inequality (19) and the LHS of Inequality (10) for given x, respectively. We consider two cases depending on whether task τ_k is active at $t_0 - x$ or not.

Before investigating each case, we first observe when task τ_k contributes to (B_x) through $Z_\theta(t_0-y)$, $W_\theta(t_0-y)$ and $N_\theta(t_0-y)$. Task τ_k contributes to (B_x) through exactly one $N_\theta(t_0-x)$-term if it has an active job at t_0-x, and there is no contribution through any $N_\theta(t_0-x)$-term if it has no active job at t_0-x. Since $Z_\theta(t_0-y)$ denotes the number of tasks whose jobs are released at t_0-y+1 with a laxity of θ, task τ_k contributes to (B_x) through $Z_\theta(t_0-y)$-terms only when its job is released at t_0-y+1, and then θ is always $D_k - C_k$. Similarly, since $W_\theta(t_0-y)$ denotes the number of tasks whose jobs are finished at t_0-y+1 and have a laxity of θ at t_0-y, task τ_k contributes to (B_x) through $W_\theta(t_0-y)$-terms only when it finishes execution at t_0-y+1. In addition, a job of τ_k can contribute to (B_x) through at most one $Z_\theta(t_0-y)$- and at most one $W_\theta(t_0-y)$-terms.

We now derive upper bounds of contribution of some W-, Z- and N-terms in the following two lemmas.

Lemma 8 *The sum of the contribution of the W-term of τ_k's qth job and the Z-term of τ_k's $(q+1)$th job to (B_x) is upper-bounded by zero, if the finishing time of τ_k's qth job and the release time of τ_k's $(q+1)$th job are in $[t_0-x+1, t_0)$.*

Proof Suppose that τ_k's qth job finishes its execution at $t_0 - y + 1$ ($1 \le y \le x$), and it has a laxity of θ at $t_0 - y$. And, suppose that τ_k's $(q+1)$th job released at $t_0 - y' + 1$ ($1 \le y' \le y$). Then, by definition, contribution of the W-term of τ_k's qth job and the Z-term of τ_k's $(q+1)$th job to (B_x) is $-\prod_{p=y}^{x}(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}) \cdot (y - \theta - 1)$ and $\prod_{p=y'}^{x}(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}) \cdot (y' - (D_k - C_k) - 1)$, respectively. Then, the sum of contribution of the W-term of τ_k's qth job and the Z-term of τ_k's $(q+1)$th job to (B_x) is calculated by

$$\begin{aligned}
&\prod_{p=y}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot (-y + \theta + 1) + \prod_{p=y'}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot \left(y' - (D_k - C_k) - 1\right) \\
&\quad = \prod_{p=y}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot \left(\theta - \prod_{p=y'}^{y-1}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot (D_k - C_k)\right) \\
&\qquad + \prod_{p=y}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot \left(-(y-1) + \prod_{p=y'}^{y-1}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot \left(y' - 1\right)\right). \qquad (39)
\end{aligned}$$

Once we apply $\theta \le D_k - C_k$ and $\prod_{p=y'}^{y-1}(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}) \ge 1$, we derive that $(\theta - \prod_{p=y'}^{y-1}(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}) \cdot (D_k - C_k))$ is upper-bounded by zero. Once we apply $\prod_{p=y'}^{y-1}(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}) \le \frac{y'}{y'-1} \cdot \frac{y'+1}{y'} \cdot \cdots \cdot \frac{y-1}{y-2} = \frac{y-1}{y'-1}$, we derive that $(-(y-1) + \prod_{p=y'}^{y-1}(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}) \cdot (y' - 1))$ is also upper-bounded by zero. Therefore, the RHS of Eq. (39) is upper-bounded by zero. □

Lemma 9 *The sum of the contribution of the N-term of τ_k's qth job, the W-term of τ_k's qth job, and the Z-term of τ_k's $(q+1)$th job to (B_x) is upper-bounded by $x - D_k + C_k$, if τ_k's qth job is active at $t_0 - x$, the finishing time of τ_k's qth job is in $[t_0 - x + 1, t_0)$, and the release time of τ_k's $(q+1)$th job is in $[t_0 - x + 1, t_0)$.*

Proof Assume that τ_k's qth job has a laxity of θ' and θ at $t_0 - x$ and $t_0 - y$ ($1 \le y \le x$), respectively, and it finishes its execution at $t_0 - y + 1$. And, assume that τ_k's $(q+1)$th job released at $t_0 - y' + 1$ ($1 \le y' \le y$). By definition, contribution of the N-term of τ_k's qth job amounts to $x - \theta'$. Then, the sum of contribution of the N-term and W-term of τ_k's qth job and the Z-term of τ_k's $(q+1)$th job to (B_x) amounts to $x - \theta'$ plus the RHS of Eq. (39) as follows:

$$\begin{aligned}
&x - \theta' + \prod_{p=y}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot \theta - \prod_{p=y'}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot (D_k - C_k) \\
&\qquad + \prod_{p=y}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot \left(-(y-1) + \prod_{p=y'}^{y-1}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot \left(y' - 1\right)\right) \\
&\quad \le x - \theta' + \prod_{p=y}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot \theta - \prod_{p=y'}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) \cdot (D_k - C_k). \qquad (40)
\end{aligned}$$

Once we apply $\theta \le \theta' \le D_k - C_k$ to the RHS of Eq. (40), we can derive the following conditions:

$$
\begin{aligned}
&\text{The RHS of Eq. (40)} \\
&\le x - \theta + \prod_{p=y}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot\theta - \prod_{p=y'}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right)\cdot(D_k - C_k) \\
&= x - D_k + C_k + \left(\prod_{p=y}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) - 1\right)\cdot\theta \\
&\quad - \left(\prod_{p=y'}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) - 1\right)\cdot(D_k - C_k) \\
&\le x - D_k + C_k + \left(\prod_{p=y}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) - 1\right)\cdot(D_k - C_k) \\
&\quad - \left(\prod_{p=y'}^{x}\left(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}\right) - 1\right)\cdot(D_k - C_k). \qquad (41)
\end{aligned}
$$

Since $\prod_{p=y'}^{x}(1+\frac{\mathbf{1}_{t_0-p}}{p-1}) \ge \prod_{p=y}^{x}(1+\frac{\mathbf{1}_{t_0-p}}{p-1})$, the RHS of Eq. (41) is upper-bounded by $x - D_k + C_k$, and this proves the lemma. □

We now prove Lemma 4 by proving (A_x) is larger than or equal to (B_x) for a given x. To do this, we use Lemmas 8 and 9, and the fact that the contribution of each W-term is no larger than zero.

Suppose that τ_k's qth job is released strictly before $t_0 - x$, and τ_k's $(q+1)$th job is released at $t_0 - y$ ($\ge t_0 - x$). And, τ_k's pth job is released strictly before t_0, and τ_k's $(p+1)$th job is released at $t_0 + v$ ($\ge t_0$). Then, among jobs invoked by τ_k, only the $(q+1)$th, $(q+2)$th, …, and pth jobs contribute to (B_x) through their Z-terms. And the $(q+1)$th, $(q+2)$th, …, and $(p-1)$th jobs of τ_k definitely contribute to (B_x) through their W-terms, and the qth and pth jobs of τ_k may or may not contribute to (B_x) through their W-terms depending on their finishing times. We now consider two cases as follows.

(Case A: task τ_k's qth job is not active at $t_0 - x$)

Since there is no active job of τ_k at $t_0 - x$, jobs of τ_k cannot contribute to (B_x) through any N-term. By Lemma 8, the contribution of a job of task τ_k through its W-term and the next job of task τ_k through its Z-term to (B_x) is upper-bounded by zero. Therefore, the contribution of the $(q+1)$th, …, and $(p-1)$th jobs of τ_k through their W-terms and the $(q+2)$th, …, and pth jobs of τ_k through their Z-terms is also upper-bounded by zero. And, the qth and pth jobs of τ_k may contribute to (B_x) through their W-terms, but any contribution though W-terms is non-positive. What remains is to calculate the contribution of $(q+1)$th jobs to (B_x) through its Z-term.

The contribution of task τ_k's $(q+1)$th job to (B_x) through its Z-term is $\prod_{p=y}^{x}(1 + \frac{\mathbf{1}_{t_0-p}}{p-1})\cdot(y - (D_k - C_k) - 1)$. Since $\prod_{p=y}^{x}(1+\frac{\mathbf{1}_{t_0-p}}{p-1})$ is upper-bounded by $\frac{y}{y-1}\cdot\frac{y+1}{y}\cdot$

$\cdots \cdot \frac{x}{x-1} = \frac{x}{y-1}$, $\prod_{p=y}^{x}(1 + \frac{\mathbf{1}_{t_0-p}}{p-1}) \cdot (y - (D_k - C_k) - 1)$ is also upper-bounded by $(y - (D_k - C_k) - 1) \cdot \frac{x}{y-1} = x - (D_k - C_k)\frac{x}{y-1} \leq x - D_k + C_k$. Therefore, the total contribution of jobs of task τ_k to (B_x) is upper-bounded by $x - D_k + C_k$.

By definition of $\delta_k^*(\theta, x)$, it holds that $\theta \leq D_k - C_k$, and there exists only one θ that results in $\delta_k^*(\theta, x) = 1$ for a given x. Thus, the contribution of jobs of τ_k to (A_x) is at least $x - D_k + C_k$.

Therefore, the total contribution of jobs of τ_k to (A_x) is larger than or equal to that to (B_x).

(Case B: task τ_k's qth job is active at $t_0 - x$ (i.e., an N-term contributes to (B_x).))

This case implies τ_k's qth job is finished after $t_0 - x$, which means it contributes to (B_x) through its W-term. We consider two sub-cases: (Case B-1) the execution of τ_k's qth job is finished strictly before t_0; and (Case B-2) at or after t_0.

(Case B-1) By Lemma 9, the contribution of the qth job through its N- and W-terms and the $(q+1)$th job through its Z- term is upper-bounded by $x - D_k + C_k$. By Lemma 8, the contribution of the $(q+1)$th, ..., and $(p-1)$th jobs of τ_k through their W-terms and the $(q+2)$th, ..., and pth jobs of τ_k through their Z-terms is also upper-bounded by zero. Here the pth job of τ_k may contributes to (B_x) through its W-term, but the quantity is no larger than zero. Therefore, the total contribution of jobs of τ_k to (B_x) is upper-bounded by $x - D_k + C_k$, and as shown in (Case 1) this means the total contribution of jobs of τ_k to (A_x) is larger than or equal to that to (B_x).

(Case B-2) In this case, the N-term of the qth job of τ_k is the only term that contributes (B_x) by jobs of τ_k; any other jobs of τ_k cannot contribute to (B_x) through any other terms. Suppose that the qth job has a laxity of θ at $t_0 - x$, then the contribution through its N-term is $x - \theta$. Since the finishing time of the qth job is at or after t_0, the release time of the job, $t_0 - z$, is no earlier than $t_0 - D_k$ (i.e., $t_0 - z \geq t_0 - D_k \iff z \leq D_k$). Then, $t_0 - x$ is at $x + D_k - z$ time units ahead of its deadline.

On the other hand, if $\delta_k(\theta', x) = 1$, then $\theta' \leq \theta$ because $x \leq x + (D_k - z)$. Therefore, the contribution of the job to (A_x), $x - \theta'$, is no smaller than $x - \theta$. Therefore, the total contribution of task τ_k to (A_x) is larger than or equal to that to (B_x).

By (Case A) and (Case B), the contribution of jobs of τ_k to (A_x) is larger than or equal to that to (B_x).

References

Anderson JH, Srinivasan A (2000) Early-release fair scheduling. In: Proceedings of Euromicro conference on real-time systems (ECRTS), pp 35–43

Andersson B, Bletsas K (2008) Sporadic multiprocessor scheduling with few preemptions. In: Proceedings of Euromicro conference on real-time systems (ECRTS), pp 243–252

Andersson B, Tovar E (2006) Multiprocessor scheduling with few preemptions. In: Proceedings of IEEE international conference on embedded and real-time computing systems and applications (RTCSA), pp 322–334

Andersson B, Baruah S, Jonsson J (2001) Static-priority scheduling on multiprocessors. In: Proceedings of IEEE real-time systems symposium (RTSS), pp 193–202

Andersson B, Bletsas K, Baruah S (2008) Scheduling arbitrary-deadline sporadic task systems on multiprocessor. In: Proceedings of IEEE international conference on embedded and real-time computing systems and applications (RTCSA), pp 197–206

Baker TP (2005) Comparison of empirical success rates of global vs. partitioned fixed-priority and EDF scheduling for hand real time. Technical Report TR-050601, Dept. of Computer Science, Florida State University, Tallahassee

Baker TP, Cirinei M (2006) A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks. In: Proceedings of IEEE real-time systems symposium (RTSS), pp 178–190

Baker TP, Cirinei M, Bertogna M (2008) EDZL scheduling analysis. Real-Time Syst 40:264–289

Baruah S, Cohen NK, Plaxton CG, Varvel DA (1996) Proportionate progress: a notion of fairness in resource allocation. Algorithmica 15(6):600–625

Baruah S, Bonifaci V, Marchetti-Spaccamela A, Stiller S (2009) Implementation of a speedup-optimal global EDF schedulability test. In: Proceedings of Euromicro conference on real-time systems (ECRTS), pp 259–268

Bertogna M, Cirinei M, Lipari G (2005) Improved schedulability analysis of EDF on multiprocessor platforms. In: Proceedings of Euromicro conference on real-time systems (ECRTS), pp 209–218

Bertogna M, Cirinei M, Lipari G (2009) Schedulability analysis of global scheduling algorithms on multiprocessor platforms. IEEE Trans Parallel Distrib Syst 20:553–566

Burns A, Baruah S (2008) Sustainability in real-time scheduling. J Comput Inf Sci Eng 2(1):74–97

Cho S, Lee S-K, Ahn S, Lin K-J (2002) Efficient real-time scheduling algorithms for multiprocessor systems. IEICE Trans Commun E85-B(12):2859–2867

Cho H, Ravindran B, Jensen ED (2006) An optimal real-time scheduling algorithm for multiprocessors. In: Proceedings of IEEE real-time systems symposium (RTSS), pp 101–110

Davis R, Burns A (2011) Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. Real-Time Syst 47(1):1–40

Davis RI, Burns A (2011) FPZL schedulability analysis. In: Proceedings of IEEE real-time technology and applications symposium (RTAS), pp 245–256

Davis RI, Burns A (2011) A survey of hard real-time scheduling for multiprocessor systems. ACM Comput Surv 43(4):35

Dertouzos ML, Mok AK (1989) Multiprocessor on-line scheduling of hard-real-time tasks. IEEE Trans Softw Eng 15:1497–1506

Easwaran A, Shin I, Lee I (2009) Optimal virtual cluster-based multiprocessor scheduling. Real-Time Syst 43(1):25–59

Fisher N, Goossens J, Baruah S (2010) Joël goossens, and sanjoy baruah. Optimal online multiprocessor scheduling of sporadic real-time tasks is impossible. Real-Time Syst 45:26–71

Funaoka K, Kato S, Yamasaki N (2008) Work-conserving optimal real-time scheduling on multiprocessors. In: Proceedings of Euromicro conference on real-time systems (ECRTS), pp 13–22

Kalyanasundaram B, Pruhs KR, Torng E (2000) Errata: A new algorithm for scheduling periodic, real-time tasks. Algorithmica 28:269–270

Lee SK (1994) On-line multiprocessor scheduling algorithms for real-time tasks. In: IEEE region 10's ninth annual international conference, pp 607–611

Lee J, Easwaran A, Shin I (2010) LLF schedulability analysis on multiprocessor platforms. In: Proceedings of IEEE real-time systems symposium (RTSS), pp 25–36

Lee J, Easwaran A, Shin I, Lee I (2010) Multiprocessor real-time scheduling considering concurrency and urgency. ACM SIGBED Rev 7(1)

Lee J, Easwaran A, Shin I (2011) Maximizing contention-free executions in multiprocessor scheduling. In: Proceedings of IEEE real-time technology and applications symposium (RTAS), pp 235–244

Lee J, Easwaran A, Shin I, Lee I (2011) Zero-laxity based real-time multiprocessor scheduling. J Syst Softw 84(12):2324–2333

Leung JY-T (1989) A new algorithm for scheduling periodic, real-time tasks. Algorithmica 4:209–219

Leung JYT, Whitehead J (1982) On the complexity of fixed-priority scheduling of periodic real-time tasks. Perform Eval 2:237–250

Levin G, Funk S, Sadowski C, Pye I, Brandt S (2010) DP-FAIR: A simple model for understanding optimal multiprocessor scheduling. In: Proceedings of Euromicro conference on real-time systems (ECRTS), pp 3–13

Liu CL, Layland J (1973) Scheduling algorithms for multi-programming in a hard-real-time environment. J ACM 20(1):46–61

Mok A (1983) Fundamental design problems of distributed systems for the hard-real-time environment. PhD thesis, Massachusetts Institute of Technology

Park M, Han S, Kim H, Cho S, Cho Y (2005) Comparison of deadline-based scheduling algorithms for periodic real-time tasks on multiprocessor. IEICE Trans Inf Syst E 88-D:658–661

Phillips CA, Stein C, Torng E, Wein J (2002) Optimal time-critical scheduling via resource augmentation. Algorithmica 32:163–200

Srinivasan A, Baruah S (2002) Deadline-based scheduling of periodic task systems on multiprocessors. Inf Process Lett 84(2):93–98