Time-Reversibility for Real-Time Scheduling on Multiprocessor Systems

Jinkyu Lee, *Member, IEEE*

Abstract—The real-time systems community has widely studied real-time scheduling, focusing on how to guarantee schedulability (i.e., timely execution) of a set of real-time tasks. However, there still exist a number of task sets that are actually schedulable by a target scheduling algorithm, but proven schedulable by none of existing schedulability tests, especially on a multiprocessor. In this paper, we propose a new paradigm for real-time scheduling, called *time-reversibility*, which views real-time scheduling *under a change in the sign of time*, and present how to utilize the paradigm for schedulability improvement. To this end, we first define the notion of a *time-reversed scheduling algorithm* and a *time-reversible schedulability test*; for example, the time-reversed scheduling algorithm against EDF (Earliest Deadline First) is LRF (Latest Release-time First). Then, we develop time-reversibility theories for schedulability improvement, which utilizes the definitions so as to *compose schedulability*. Finally, we generalize the definitions and theories to job-level dynamic-priority scheduling in which the priority of a job may vary with time, such as EDZL (Earliest Deadline first until Zero Laxity). Specifically, we accommodate time-varying job parameters to the time-reversibility definitions, and adapt the time-reversibility theories for the additional necessary deadline-miss conditions specialized for a class of job-level dynamic-priority scheduling algorithms. As case studies, we demonstrate that the time-reversibility theories help to find up to 13.6% additional EDF- and EDZL-schedulable task sets.

Index Terms—Real-time scheduling, schedulability analysis, time-reversibility

1 Introduction

The real-time systems community has addressed how to guarantee timely execution of real-time tasks, by developing scheduling algorithms and their schedulability tests. A scheduling algorithm decides the order of execution of jobs periodically/sporadically invoked by a set of real-time tasks, and its schedulability test judges whether all the jobs scheduled by the algorithm finish their executions within their deadlines.

Although multiprocessor systems have become popular due to its potential for high performance at low cost, the real-time scheduling theories for the systems have a long way to go. For example, no exact (i.e., sufficient and necessary) schedulability test that exhibits polynomial time-complexity has been developed on a multiprocessor even for the most popular preemptive scheduling algorithms: EDF (Earliest Deadline First) and RM (Rate Monotonic) [1].[1] Instead, different sufficient schedulability tests have been developed, aiming at finding additional schedulable task sets that are not deemed schedulable by any existing schedulability tests. Although useful in covering additional task sets, all the existing schedulability tests have shared the common principle—investigating real-time scheduling *as it is in terms of a time order*.

• *J. Lee is with Department of Computer Science and Engineering, Sungkyunkwan University (SKKU), Republic of Korea.*
E-mail: jinkyu.lee@skku.edu

1. There are some exact schedulability tests with exponential time-complexity [2, 3].

In this paper, we propose a new paradigm for real-time scheduling, called *time-reversibility*, and exploit the paradigm for schedulability improvement. Different from existing studies that focus on scheduling of a series of jobs in a time-ordered manner, we view real-time scheduling *by tracing back to time*, i.e., under a change in the sign of time. To this end, we construct a job J_i^{-q} that corresponds to a given job J_i^q as follows: (i) J_i^{-q}'s deadline is set to J_i^q's release time under a change in the plus-minus sign, (ii) J_i^{-q}'s release time is set to J_i^q's deadline under a change in the sign, and (iii) the priority of J_i^{-q} is set to that of J_i^q. Fig. 1 shows an example; since the release time and deadline of J_i^2 are 10 and 17, respectively, the release time and deadline of J_i^{-2} are -17 and -10, respectively. Then, for a given scheduling algorithm G that prioritizes $\{J_i^q\}$, a scheduling algorithm that prioritizes $\{J_i^{-q}\}$ is said to be a time-reversed scheduling algorithm against G (denoted by G^-). For example, since EDF gives the highest priority to a job with the *earliest deadline*, a time-reversed scheduling algorithm against EDF is LRF (Latest Release-time First), which assigns the highest priority to a job with the *latest release time*; the converse also holds.

For a connection between a time-reversed scheduling algorithm against G (i.e., G^-) and a schedulability test A_G for G, we define the notion of time-reversibility of A_G with respect to task-set-, task-, and execution-level schedulability. For example, a schedulability test A_G for a scheduling algorithm G is said to be *time-reversible with respect to task-set-level schedulability*, if all task sets deemed schedulable by A_G are also schedulable by G^-. And, a schedulability test for a scheduling algorithm G is said to be *time-reversible with respect to execution-level schedulability*, if the following

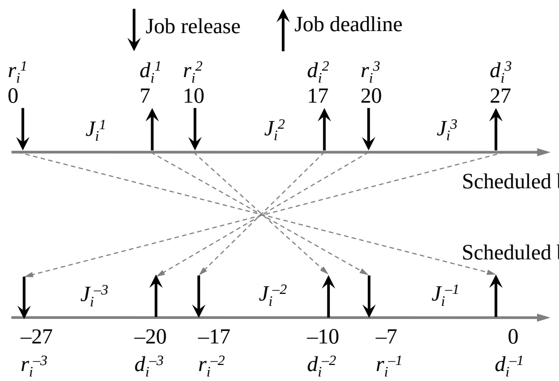


Fig. 1. Time-reversibility for real-time scheduling: jobs under a scheduling algorithm G and the corresponding jobs under its time-reversed scheduling algorithm G^-

statements holds: if the test guarantees that every job of a task under G executes X time units between its release time and the release time after ℓ time units, it is guaranteed that every job of the task under G^- executes X time units between its deadline ahead of ℓ time units and the deadline (see Fig. 2). As an example, we focus on EDF and its time-reversed scheduling algorithm LRF, and prove that RTA (Response-Time Analysis) and DA (Deadline Analysis) for EDF (developed in [4, 5]) and those for LRF (developed in this paper) are time-reversible with respect to all the three levels of schedulability.

To utilize the notion of time-reversibility for schedulability improvement, we can exploit the time-reversibility definition *as it is*. That is, provided that a schedulability test A_G of G is time-reversible with respect to task-set-level schedulability, all task sets deemed schedulable by A_G are actually schedulable by G^-, potentially finding additional task sets schedulable by G^-. For example, we show that RTA for LRF (developed in this paper) finds additional EDF-schedulable task sets that are not deemed schedulable by any existing EDF schedulability tests.

Beyond simple application of the time-reversibility definitions, we further improve schedulability by *composing schedulability* using the notion of time-reversibility with respect to task- and execution-level schedulability. For example, a task' schedulability under G^- can be composed by two schedulability tests: (i) the first some execution directly guaranteed by a schedulability test for G^-, and (ii) the remaining execution indirectly guaranteed by an execution-level time-reversible schedulability test for G. As a case study, we demonstrate that a collaboration between RTA for EDF and RTA for LRF results in covering additional EDF-schedulable task sets, which are not deemed schedulable by any single schedulability tests including themselves.

While the above time-reversibility definitions and theories are confined to job-level fixed-priority scheduling, we want to make them applicable even to job-level dynamic-priority scheduling in which a priority of the same job may vary with time. To this end, we generalize the time-reversibility definitions by accommodating time-varying job parameters. Then, we target a class of job-level dynamic-priority scheduling algorithms called ZL-based (zero-laxity) scheduling algorithms [6], which give the highest priority to jobs with the zero-laxity state, where a laxity of a job at an instant is defined as the difference between the time to its deadline and the remaining execution at the instant. The ZL-based scheduling algorithms have an additional necessary deadline-miss condition; for a deadline miss, there should be at least $m+1$ tasks that are capable of reaching the zero-laxity state, where m is the number of processors. By accommodating the necessary deadline-miss condition, we adapt the time-reversibility theories for ZL-based scheduling algorithms, and demonstrate the application to a popular ZL-based scheduling algorithm EDZL (Earliest Deadline first until Zero Laxity) [7], which gives the highest-priority to zero-laxity jobs and schedules other jobs by EDF.

We then demonstrate via extensive simulation that our new EDF schedulability test derived from the time-reversibility theories can find up to 13.6% additional EDF-schedulable task sets that are not covered by the best existing EDF schedulability test on a multiprocessor platform. We also show that our new EDZL schedulability test covers additional EDZL-schedulable task sets.

In summary, this paper makes the following contributions:

- Proposal of the new paradigm for real-time scheduling, called time-reversibility,
- Establishment of the theoretical foundation of time-reversibility for schedulability improvement,
- Application of the time-reversibility theories to a popular scheduling algorithm EDF, demonstrating the effectiveness of the notion in improving schedulability,
- Generalization of the time-reversibility definitions for job-level dynamic-priority scheduling and adaptation of the time-reversibility theories for EDZL, demonstrating their wide applicability, and
- Substantiation of quantitative schedulability improvement through extensive simulation.

The rest of this paper is structured as follows. Section 2 describes our systems model and notations. Section 3 gives formal definitions of time-reversibility of a schedulability test, and perform case studies for EDF and LRF schedulability tests. Section 4 develops time-reversibility theories towards schedulability improvement, and applies the theories to EDF schedulability tests. Section 5 generalizes the time-reversibility definitions and theories to job-level dynamic-priority scheduling, with a case study for EDZL. Section 6 evaluates the new schedulability tests developed in this paper, in terms of schedulability improvement and time-complexity. Finally, this paper concludes with Section 7.

2 System Model and Notations

We consider a task set τ consisting of $|\tau|$ sporadic real-time tasks $\tau_i(T_i, C_i, D_i)$, where T_i is the minimum separation, C_i is the worst-case execution time, and D_i is the relative deadline [8]. We focus on implicit- and constrained-deadline tasks, which satisfy $D_i = T_i$ and $D_i \le T_i$, respectively.

For convenience' sake, we assume a quantum-based time with the quantum length equal to one time unit, without loss of generality. All task parameters are multiples of the quantum.

A task τ_i invokes a series of sporadic jobs, each separated from its predecessor by at least T_i time units. Each job of τ_i, once released, should finish its execution within D_i time units. The q^{th} job of τ_i is denoted by J_i^q, and the release time and deadline of J_i^q are denoted by r_i^q and d_i^q, respectively (where $d_i^q = r_i^q + D_i$).

In this paper, we consider a multiprocessor computing platform consisting of m identical processors, where m is an integer value. For the ease of presentation, we will not specify the computing platform when no ambiguity arises in the rest of the paper.

When it comes to scheduling algorithms, this paper focuses on scheduling algorithms that are *global*, *preemptive*, and *work-conserving*. That is, a job can execute at any core (*global*); a higher-priority job can preempt a lower-priority job at any time (*preemptive*); and no processor can be left idle as long as there is an unfinished job in the system (*work-conserving*).

A schedulability test A_G for a target scheduling algorithm G judges schedulability of a task or a task set under G, defined as follows. A task $\tau_k \in \tau$ is said to be *schedulable* by G, if no job invoked by τ_k triggers the first deadline miss when τ with any legal job arrival sequence is scheduled by G [9]. Also, τ is said to be *schedulable* by G, if every task τ_k belonging to τ is schedulable by G.

3 Time-Reversibility Definitions

As a first step to exploit the notion of time-reversibility towards schedulability improvement, this section presents time-reversibility definitions for real-time scheduling. To this end, this section introduces the notion of a *time-reversed scheduling algorithm*. Followed by the notion, the section gives a formal definition of a *time-reversible schedulability test*, with respect to three different levels of schedulability. Finally, the section checks time-reversibility of (i) existing schedulability tests for EDF and (ii) newly-developed ones for LRF (i.e., a time-reversed scheduling algorithm against EDF).

3.1 Definition of a time-reversed scheduling algorithm

Suppose that a series of jobs invoked by τ (denoted by $\{J_i^q\}_{\tau_i \in \tau}$) is executed by a scheduling algorithm G. We now look at $\{J_i^q\}_{\tau_i \in \tau}$ *under a change in the sign of time*. To this end, we synthesize another series of jobs (denoted by $\{J_i^{-q}\}_{\tau_i \in \tau}$), which is a one-to-one mapping of $\{J_i^q\}_{\tau_i \in \tau}$ as follows.

R$_1$. The release time of J_i^{-q} (i.e., r_i^{-q}) is set to $-d_i^q$, and the deadline of J_i^{-q} (i.e., d_i^{-q}) is set to $-r_i^q$; recall that d_i^q and r_i^q denote the deadline and the release time of J_i^q, respectively.

R$_2$. The worst-case execution time of J_i^{-q} is set to that of J_i^q.

R$_3$. The priority of J_i^{-q} is set to that of J_i^q.

For example, since the release time of J_i^2 in Fig. 1 is $r_i^2 = 10$, the deadline of J_i^{-2} (corresponding to J_i^2) is $d_i^{-2} = -10$. Likewise, provided that the deadline of J_i^2 in the same figure is $d_i^2 = 17$, the release time of J_i^{-2} is $r_i^{-2} = -17$.

Note that $\{J_i^{-q}\}_{\tau_i \in \tau}$ is also an instance of a series of jobs invoked by τ in that it conforms with all the task parameters of τ. We also note that R$_1$–R$_3$ provide mapping of static job-parameters only (e.g., the release time, the deadline and the worst-case execution time of a job), which are components of job-level fixed-priority scheduling algorithms in which the priority of a job cannot change over time. Section 5 will generalize them for job-level dynamic-priority scheduling algorithms.

If we pay attention to two corresponding job-level fixed-priority scheduling algorithms that prioritize $\{J_i^q\}_{\tau_i \in \tau}$ and $\{J_i^{-q}\}_{\tau_i \in \tau}$, respectively, there is a relationship between the two, defined as follows.

Definition 1. Suppose that for a given $\{J_i^q\}_{\tau_i \in \tau}$ which is prioritized by a *job-level fixed-priority* scheduling algorithm G, $\{J_i^{-q}\}_{\tau_i \in \tau}$ is generated according to R$_1$–R$_3$. Then, we can derive a corresponding scheduling algorithm G^-, such that G^- directly assigns job priorities to $\{J_i^{-q}\}_{\tau_i \in \tau}$. A scheduling algorithm G^- is said to be a *time-reversed scheduling algorithm* against G.

Here we present two examples of G^- for a given G.

Example 3.1. Since J_i^q's deadline matches J_i^{-q}'s release time under a change in the plus-minus sign, scheduling of $\{J_i^q\}_{\tau_i \in \tau}$ by EDF (that gives the highest priority to a job with the *earliest deadline*) corresponds to that of $\{J_i^{-q}\}_{\tau_i \in \tau}$ by a scheduling algorithm that gives the highest priority to a job with the *latest release time*, which is called LRF (Latest Release-time First). In other words, LRF is a time-reversed scheduling algorithm against EDF (denoted by LRF = EDF$^-$). Similarly, EDF = LRF$^-$ holds.

Example 3.2. Scheduling of $\{J_i^q\}_{\tau_i \in \tau}$ by RM corresponds that of $\{J_i^{-q}\}_{\tau_i \in \tau}$ by the same scheduling algorithm RM because the priority of a job does not depend on its release time and deadline. In other words, RM = RM$^-$ holds. The same relationship holds for DM (Deadline Monotonic) [10].

3.2 Definition of a time-reversible schedulability test

Since we are interested in schedulability guarantees, we need to establish a relationship between a schedulability test A_G for a scheduling algorithm G and its time-reversed scheduling algorithm G^- in terms of schedulability. Based on the notion of a time-reversed scheduling algorithm, we provide formal definitions of a time-reversible schedulability test as for three different levels of schedulability, recorded in the following definition.

Definition 2. A schedulability test A_G for a scheduling algorithm G is said to be *time-reversible* with respect to

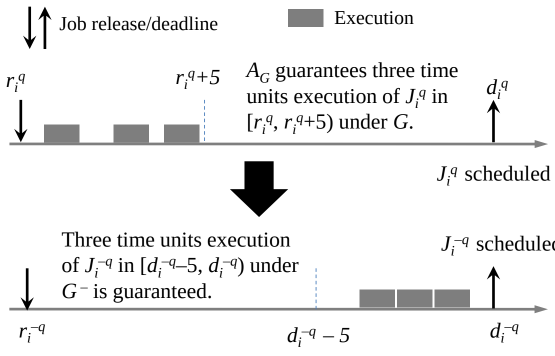

Job release/deadline
Execution
r_i^q
r_i^q+5
A_G guarantees three time units execution of J_i^q in $[r_i^q, r_i^q+5)$ under G.
d_i^q
J_i^q scheduled by G
Three time units execution of J_i^{-q} in $[d_i^{-q}-5, d_i^{-q})$ under G^- is guaranteed.
J_i^{-q} scheduled by G^-
r_i^{-q}
$d_i^{-q}-5$
d_i^{-q}

Fig. 2. Time-reversibility of a schedulability test with respect to execution-level schedulability

- *task-set-level schedulability*, if the following statement holds for every τ:
  - if τ is deemed schedulable by A_G, τ is also schedulable by G^-;
- *task-level schedulability*: if the following statement holds for every $\tau_i \in \tau$:
  - if $\tau_i \in \tau$ is deemed schedulable by A_G, $\tau_i \in \tau$ is also schedulable by G^-; and
- *execution-level schedulability*, if the following statement holds for every $\tau_i \in \tau$, $C'_i \in [0, C_i]$, and $\ell \in [0, D_i]$:
  - if A_G guarantees that the amount of execution of every job of $\tau_k \in \tau$ under G (denoted by J_k^q) performed in $[r_k^q, r_k^q + \ell)$ is C'_k, that of every job of $\tau_k \in \tau$ under G^- (denoted by J_k^{-q}) performed in $[d_k^{-q} - \ell, d_k^{-q})$ is equal to either (a) at least C'_k if the amount of the remaining execution of J_k^{-q} at $d_k^{-q} - \ell$ is no smaller than C'_k or (b) the amount of the remaining execution of J_k^{-q} at $d_k^{-q} - \ell$ otherwise.

Fig. 2 illustrates time-reversibility of a schedulability test with respect to execution-level schedulability.

Then, there exist relationships between the above time-reversibility definitions, as stated in the following lemma.

Lemma 1. The following inclusive relationship holds among the three time-reversibility definitions of a schedulability test A_G for a scheduling algorithm G.

I$_1$. If A_G is time-reversible with respect to task-level schedulability, then it is also time-reversible with respect to task-set-level schedulability.

I$_2$. If A_G is time-reversible with respect to execution-level schedulability, then it is also time-reversible with respect to task- and task-set-level schedulability.

Proof: By the definition of the schedulability of a task and a task set in Section 2, I$_1$ holds immediately.

If we substitute C'_i with C_i and ℓ with D_i, the time-reversibility definition with respect to execution-level schedulability is equivalent to that with respect to task-level schedulability. Then, by I$_1$, the remaining part of I$_2$ holds. □

3.3 Time-reversibility of EDF schedulability tests

This subsection discovers time-reversibility of existing EDF schedulability tests. To this end, we first recapitulate popular schedulability test frameworks for EDF, and then prove their time-reversibility.

In order to judge whether every job invoked by a set of real-time tasks finishes its execution within its deadline, many schedulability test frameworks have been developed. Among the frameworks, RTA (Response-Time Analysis) [4] and DA (Deadline Analysis) [5, 11] have been popular due to their applicability and schedulability performance.

RTA focuses on a job of interest of τ_k (called J_k^q) and calculates the length of cumulative intervals in $[r_k^q, r_k^q + \ell)$ such that jobs of τ_i execute while J_k^q cannot, where $0 < \ell \le D_k$. This is called interference of τ_i on τ_k in an interval $[r_k^q, r_k^q + \ell)$, and denoted by $I_{k\leftarrow i}(r_k^q, r_k^q + \ell)$. Since a job cannot execute in a time slot only when other m higher-priority jobs execute, $\frac{1}{m}\sum_{\tau_i \in \tau - \{\tau_k\}} I_{k\leftarrow i}(r_k^q, r_k^q + \ell)$ represents the length of cumulative intervals in $[r_k^q, r_k^q + \ell)$ such that J_k^q cannot execute due to other jobs' execution. Therefore, if the value is no larger than $\ell - C_k$, the job J_k^q finishes its full execution (as much as C_k) at or before $r_k^q + \ell$. Using the notion of interference, RTA judges the schedulability of a task as follows.

Lemma 2 (RTA: Theorem 3 in [4]). A task $\tau_k \in \tau$ is schedulable, if every job J_k^q invoked by τ_k satisfies Eq. (1) for some $C_k \le \ell \le D_k$:

$$C_k + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau - \{\tau_k\}} \min\Big(I_{k\leftarrow i}(r_k^q, r_k^q + \ell), \ell - C_k + 1\Big) \right\rfloor \le \ell. \tag{1}$$

Proof: Here, we summarize the proof in [4]. Since a job cannot execute in a time slot only when other m higher-priority jobs execute, $X \overset{\mathsf{def.}}{=} C_k + \left\lfloor \frac{1}{m}\sum_{\tau_i \in \tau - \{\tau_k\}} I_{k\leftarrow i}(r_k^q, r_k^q + \ell) \right\rfloor$ represents the duration between J_k^q's release and finishing time. By the definition of $I_{k\leftarrow i}(r_k^q, r_k^q + \ell)$, if $I_{k\leftarrow i}(r_k^q, r_k^q + \ell) > \ell - C_k + 1$ holds for some τ_i, J_k^q cannot finish its execution in $[r_k^q, r_k^q + \ell)$. Therefore, the following relation holds: if X is strictly larger than ℓ, the LHS is also strictly larger than ℓ. By the contraposition, the lemma holds. □

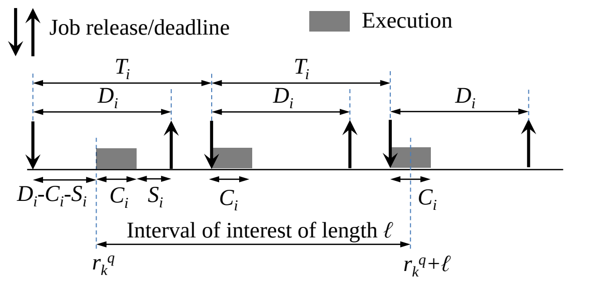
We will present how to find such ℓ later in this subsection.

Different from RTA, DA focuses only on $\ell = D_k$, as recorded in the following lemma.

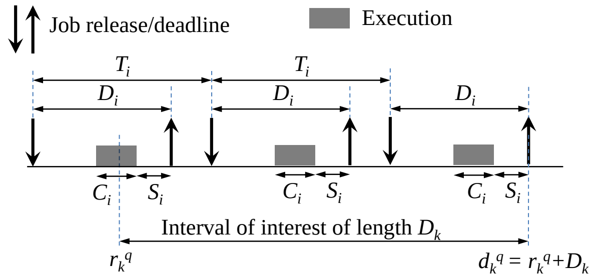
Lemma 3 (DA: Theorem 5 in [11]). A task $\tau_k \in \tau$ is schedulable, if every job J_k^q invoked by τ_k satisfies Eq. (1) for $\ell = D_k$.

Proof: Since the lemma is a special case of Lemma 2, the lemma holds. □

Since $I_{k\leftarrow i}(r_k^q, r_k^q + \ell)$ in Eq. (1) is algorithm-dependent, the main issue to develop RTA and DA for a target



(a) $W_i(\ell, S_i)$ under any work-conserving scheduling algorithm



(b) $E_i(D_k, S_i)$ under EDF

Fig. 3. Upper-bounds of interference $I_{k\leftarrow i}(r_k^q, r_k^q+\ell)$: $W_i(\ell, S_i)$ under any work-conserving scheduling algorithm and $E_i(D_k, S_i)$ under EDF

scheduling algorithm is to derive a tight upper-bound of $I_{k\leftarrow i}(r_k^q, r_k^q+\ell)$. Existing studies calculate two upper-bounds of the interference: the one commonly applied to any work-conserving scheduling algorithm and the other specialized for the target scheduling algorithm.

Since the amount of interference of τ_i on τ_k in an interval is upper-bounded by that of executions of jobs of τ_i in the interval, existing studies found when the amount of executions of jobs of τ_i is maximized in a given interval. That is, the first job of τ_i in the interval executes as late as possible and other jobs in the interval execute as early as possible; also, the interval starts when the first job starts its execution as shown in Fig. 3(a). In this situation, the number of jobs of τ_i executed in the interval except the last job, denoted by $N_i(\ell, S_i)$, is calculated as follows [4]:

$$N_i(\ell, S_i) = \left\lfloor \frac{\ell + D_i - C_i - S_i}{T_i} \right\rfloor, \tag{2}$$

where S_i denotes a lower-bound of the interval between a completion time and deadline of every job invoked by τ_i, called *slack value*. In other words, every job J_i^q of τ_i finishes its execution until $d_i^q - S_i$, and therefore does not execute in $[d_i^q - S_i, d_i^q)$.

For instance, $N_i(\ell, S_i) = 2$ holds in Fig. 3(a), and those $N_i(\ell, S_i)$ jobs of τ_i fully execute in the interval of length ℓ, contributing to $N_i(\ell, S_i) \cdot C_i$. Considering the contribution of the last job, the amount of maximum execution of jobs of τ_i in an interval of length ℓ can be calculated by $W_i(\ell, S_i)$ as follows [4]:

$$W_i(\ell, S_i) = \\ N_i(\ell, S_i) \cdot C_i + \min\Big(C_i, \ell + D_i - C_i - S_i - N_i(\ell, S_i) \cdot T_i\Big), \tag{3}$$

which is an upper-bound of $I_{k\leftarrow i}(r_k^q, r_k^q+\ell)$ for any work-conserving scheduling algorithm.

On the other hand, if we focus on an interval $[r_k^q, r_k^q+D_k)$ between a release time and deadline of J_k^q of τ_k, we can derive another upper-bound of $I_{k\leftarrow i}(r_k^q, r_k^q+D_k)$ tailored to EDF. Under EDF, a job J_i^p can interfere with another job J_k^q only when the deadline of J_i^p is no later than that of J_k^p. Therefore, $I_{k\leftarrow i}(r_k^q, r_k^q+D_k)$ under EDF is maximized when the deadline of a job of τ_i is aligned to the end of the interval, and all jobs of τ_i execute as late as possible as shown in Fig. 3(b). This is calculated by $E_i(D_k, S_i)$ [4], where

$$E_i(\ell, S_i) = \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot C_i + \max\Big(0, \min\Big(C_i, \ell - \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot T_i - S_i\Big)\Big). \tag{4}$$

Finally, by taking the minimum of the two upper-bounds, RTA for EDF uses the following upper-bound of interference $I_{k\leftarrow i}(r_k^q, r_k^q+\ell)$.

$$I_{k\leftarrow i}(r_k^q, r_k^q+\ell) \text{ under EDF with slack reclamation} \\ \leq \min\big(W_i(\ell, S_i), E_i(D_k, S_i)\big). \tag{5}$$

Then, RTA for EDF works as follows [4]. For each task $\tau_k \in \tau$, Eq. (1) with applying Eq. (5) is investigated with the initial value $\ell = C_k$. If the inequality holds, the task is deemed schedulable. Otherwise, RTA for EDF resets ℓ to the previous value of the LHS of the inequality, until the inequality holds (schedulable task) or $\ell > D_k$ (unschedulable task). If a task τ_k is deemed schedulable, the value of ℓ that satisfies the inequality is an upper-bound of the response time of τ_k (denoted by R_k).

In this process, RTA for EDF exploits slack values S_i as follows. Initially, every S_i in the LHS of Eq. (5) is set to zero, and every task's response time is calculated. Then, we reset every schedulable task's slack (i.e., S_i) to $D_i - R_i$ (if positive) and repeat to calculate every task's response time, until all tasks are deemed schedulable (schedulable task set) or there is no slack value update (unschedulable task set). This schedulability test is called *RTA for EDF with slack reclamation*.

On the other hand, we skip the repetition for slack reclamation, by statically setting all slack values to zero as recorded in Eq. (6). This schedulability test is called *RTA for EDF without slack reclamation*.

$$I_{k\leftarrow i}(r_k^q, r_k^q+\ell) \text{ under EDF without slack reclamation} \\ \leq \min\big(W_i(\ell, 0), E_i(D_k, 0)\big). \tag{6}$$

Similar to RTA for EDF, DA for EDF employs Eqs. (5) and (6) for $\ell = D_k$, yielding two different schedulability tests with/without slack reclamation [11]. Here, the slack value S_i is calculated by the difference between D_k and the LHS of Eq. (1), if Eq. (1) holds for $\ell = D_k$ (otherwise 0) [11].

From now on, we investigate time-reversibility of RTA and DA for EDF, starting from RTA for EDF without slack reclamation, as stated in the following lemma.

Lemma 4. RTA for EDF without slack reclamation (i.e., Lemma 2 with applying Eq. (6)) is time-reversible with respect to execution-, task-, and task-set-level schedulability.

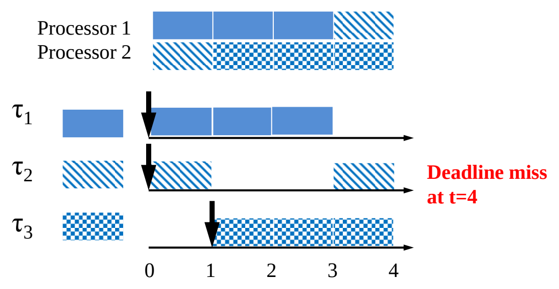


Fig. 4. A counter example of time-reversibility of RTA for EDF with slack reclamation: $\tau = \{\tau_1(4,3,4) = \tau_2, \tau_3(40,3,40)\}$ on two processors

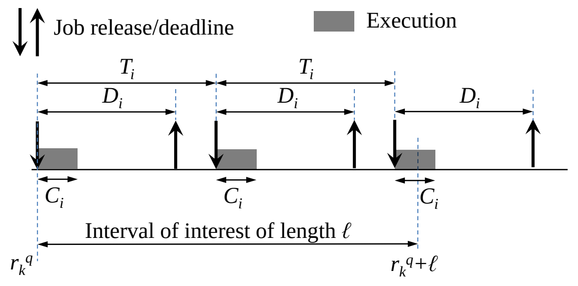


Fig. 5. An upper-bound of interference $I_{k\leftarrow i}(r_k^q, r_k^q+\ell)$ under LRF: $L_i(\ell)$

Proof: By Lemma 1, it suffices to prove the lemma for execution-level schedulability. Suppose that RTA for EDF without slack reclamation guarantees that X amount of execution is performed between each job's release time and the time after ℓ time units (i.e., $[r_k^q, r_k^q+\ell)$). Then, we prove that X amount of execution is performed between each job's deadline ahead of ℓ time units and the deadline (i.e., $[d_k^q-\ell, d_k^q)$) under LRF.

Under LRF, a job of τ_i can interfere with another job J_k^q only when the release time of the job of τ_i is no earlier than J_k^q. Therefore, the amount of interference of jobs of τ_i on J_k^q is maximized when the release time of the first job of τ_i is aligned with that of J_k^q. Then, the scenario that yields the maximum interference under LRF shown in Fig. 5 is vertically symmetrical to the scenario of $E_i(D_k,0)$ in Fig. 3(b), where $[d_k^q-D_k, d_k^q)$ in Fig. 3(b) corresponds to $[r_k^q, r_k^q+D_k)$ in Fig. 5. This means, jobs of τ_i under LRF interfere with J_k^q during at most $E_i(D_k,0)$. We also directly prove this upper-bound as follows. First, we can calculate the number of jobs of τ_i that contribute the full execution in $[r_k^q, r_k^q+D_k)$ in Fig. 5, (i.e. the first two jobs in the figure), which is $\lfloor \frac{D_k}{T_i} \rfloor$. Second, the contribution of the last job of τ_i in $[r_k^q, r_k^q+D_k)$ is $\min(C_i, D_k - \lfloor \frac{D_k}{T_i} \rfloor \cdot T_i)$ (i.e., the third job in Fig. 5). Therefore, the total interference of jobs of τ_i to J_k^q is upper-bounded by $E_i(D_k,0)$, which proves the lemma. □

Different from RTA for EDF without slack reclamation, that with slack reclamation (i.e., Lemma 2 with applying Eq. (5)) is *not* time-reversible with respect to even task-set-level schedulability, as demonstrated in the following counter example.

Example 3.3. Suppose that $\tau = \{\tau_1(4,3,4) = \tau_2, \tau_3(40,3,40)\}$ is scheduled by EDF on two processors. Then, τ is deemed schedulable by RTA for EDF with slack reclamation. However τ is indeed not schedulable by LRF (i.e., time-reversed scheduling algorithm against EDF). This is because, if τ_1 and τ_2 invoke their jobs at $t=0$ and τ_3 invokes its job at $t=1$, one of the jobs of either τ_1 or τ_2 misses its deadline since the job of τ_3 has a higher priority under LRF, as shown in Fig. 4.

We can explain non-time-reversibility of RTA for EDF with slack reclamation as follows. Since the slack implies that a job of interest J_i^q cannot execute just before its deadline under a scheduling algorithm G, it implies that the corresponding job J_i^{-q} cannot execute right after its release time under G^-, which does not necessarily hold under G^-. For example, a slack of J_i^q under EDF matches no execution right after J_i^{-q}'s release time under LRF; however, LRF itself does not prevent a job's execution right after its release time.

Similar to Lemma 4, DA for EDF without slack reclamation (i.e., Lemma 3 with applying Eq. (6)) is time-reversible with respect to task- and task-set-level schedulability. This is because, DA for EDF is a special case of RTA for EDF (i.e., applying $\ell = D_k$). Note that DA for EDF with slack reclamation (i.e., Lemma 3 with applying Eq. (5)) is *not* time-reversible with respect to even task-set-level schedulability.

3.4 Time-reversibility of new LRF schedulability tests

While the previous subsection focuses on EDF, this subsection investigates time-reversibility of schedulability tests for LRF (i.e., a time-reversed scheduling algorithm against EDF). Since no schedulability test for LRF exists so far, we first develop new LRF schedulability tests, and then investigate time-reversibility of the LRF tests. These LRF schedulability tests become a basis for improving EDF schedulability using time-reversibility theories to be presented in Section 4.

Under LRF, a job J_i^p can interfere with another job J_k^q only when the release time of J_i^p is no earlier than that of J_k^q. Therefore, $I_{k\leftarrow i}(r_k^q, r_k^q+\ell)$ is maximized when the interval of interest begins at the release time of the first job of τ_i in the interval and all jobs of τ_i in the interval execute as early as possible, as shown in Fig. 5. Then, the amount of maximum interference of jobs of τ_i on J_k^q in $[r_k^q, r_k^q+\ell)$ is calculated by $L_i(\ell)$ as follows:

$$L_i(\ell) = \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot C_i + \min\left(C_i, \ell - \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot T_i\right). \tag{7}$$

Combined with the upper-bound of interference under any work-conserving scheduling algorithm $W_i(\ell, S_i)$, the interference of τ_i on τ_k under LRF is upper-bounded as follows.

$$\begin{aligned} I_{k\leftarrow i}(r_k^q, r_k^q+\ell) \text{ under LRF} &\le \min\left(W_i(\ell, S_i), L_i(\ell)\right) \\ &= L_i(\ell). \end{aligned} \tag{8}$$

Note that since $L_i(\ell)$ does not depend on S_i, we have only one RTA for LRF (without slack reclamation), which is Lemma 2 with applying Eq. (8). When it comes to DA, DA for LRF employs Eq. (8) for $\ell = D_k$, i.e., Lemma 3 with applying Eq. (8).

Then, RTA for LRF is time-reversible as recorded in the following lemma.

Lemma 5. RTA for LRF (i.e., Lemma 2 with applying Eq. (8)) is time-reversible with respect to execution-, task-, and task-set-level schedulability.

Proof: By Lemma 1, it suffices to prove the lemma for execution-level schedulability. We prove that $L_i(\ell)$ is no larger than the amount of time in $[d_k^q - \ell, d_k^q)$ jobs of τ_i can interfere with J_k^q when the scheduling algorithm is EDF. Then, it holds that any job of τ_k under EDF does not miss its deadline as long as RTA for LRF guarantees the schedulability of τ_k.

By definition, $L_i(\ell)$ in Eq. (7) is equal to $E_i(\ell)$ with $S_i = 0$ in Eq. (4). Since $E_i(\ell)$ with $S_i = 0$ is an upper-bound of the amount of interference of jobs of τ_i on J_k^q in $[d_k^q - \ell, d_k^q)$ under EDF, the lemma holds. □

Since DA for LRF (i.e., Lemma 3 with applying Eq. (8)) is also a special case of RTA for LRF (i.e., applying $\ell = D_k$), DA for LRF is also time-reversible with respect to task- and task-set-level schedulability.

4 Time-reversibility Theories for Schedulability Improvement

While the previous section introduces formal definitions of time-reversibility of a schedulability test and discovers time-reversible schedulability tests, we need to utilize the notion of time-reversibility for schedulability improvement. To this end, this section presents how to improve schedulability using the time-reversibility definition as it is. Then, the section develops ways to compose schedulability by utilizing the definitions.

4.1 Schedulability improvement using time-reversibility definition as it is

For schedulability improvement, we directly use Definition 2, as stated in the following theorem.

Theorem 1. Suppose that a schedulability test A_G for a scheduling algorithm G is time-reversible with respect to task-set-level schedulability. Then, if A_G deems a task set τ schedulable by G, τ is schedulable by G^-.

Proof: According to Definition 2, the theorem immediately holds. □

Although straightforward, Theorem 1 can be useful in finding additional task sets schedulable by G^-, which are not deemed schedulable by any existing schedulability test for G^-. Here are two examples that demonstrate usefulness of the theorem in discovering additional schedulable task sets.

Example 4.1. Suppose that $\tau = \{\tau_1(3,1,3), \tau_2 = \tau_3 = \tau_4 = (2,1,2)\}$ is scheduled by EDF on a two-processor platform. Then, while τ is not deemed schedulable by any single existing EDF schedulability test in a survey [12], RTA for LRF (i.e., Lemma 2 with applying Eq. (8)) guarantees the task set's schedulability under EDF due to its time-reversibility proved in Lemma 5.

Example 4.2. Suppose that $\tau = \{\tau_1(2,1,2) = \tau_2 = \tau_3\}$ is scheduled by LRF on a two-processor platform. Then, τ is deemed schedulable by RTA for EDF without slack reclamation (i.e., Lemma 2 with applying Eq. (6)); by time-reversibility proved in Lemma 4, RTA for EDF without slack reclamation guarantees the task set's schedulability under LRF.
To the best knowledge of the author, no schedulability test *specialized* for LRF has been developed. Therefore, the best existing schedulability test to be applied to LRF is the state-of-the-art schedulability test for any work-conserving (WC) scheduling algorithm, which is RTA for WC with slack reclamation (i.e., Lemma 2 with applying $I_{k\leftarrow i}(r_k^q, r_k^q+\ell) \le W_i(\ell, S_i)$). However, RTA for WC with slack reclamation does not deem τ schedulable.

4.2 Schedulability composition using time-reversibility

While we can immediately improve schedulability using the definition of time-reversibility with respect to task-set-level schedulability *as it is*, we can *compose schedulability* using time-reversibility regarding task- and execution-level schedulability. The following theorem presents schedulability composition[2] using time-reversibility as for task-level schedulability.

Theorem 2. Suppose that there exist two schedulability tests, one for a scheduling algorithm G and the other for its time-reversed scheduling algorithm G^- (denoted by A_G and B_{G^-}, respectively), and A_G is *time-reversible with respect to task-level schedulability*. Then, a task set τ is schedulable by G^-, if every task $\tau_k \in \tau$ is deemed schedulable by either A_G or B_{G^-}.

Proof: By Definition 2, all tasks deemed schedulable by A_G are also schedulable by G^-. Therefore, the theorem holds. □

Beyond composition of task-set-level schedulability from individual task-level schedulability, we can compose task-level schedulability from the time-reversibility with respect to execution-level schedulability, as recorded in the following theorem.

Theorem 3. Suppose that there exist two schedulability tests, one for a scheduling algorithm G and the other for its time-reversed scheduling algorithm G^- (denoted by A_G and B_{G^-}, respectively), and A_G is *time-reversible with respect to execution-level schedulability*. Then, a task $\tau_k \in \tau$ is schedulable, if there exist $C_k' \in [0, C_k]$ and $\ell \in [0, D_k]$ such that A_G guarantees that every job of $\tau_k \in \tau$ under G (denoted by J_k^q) finishes its execution at least as much as C_k' in $[r_k^q, r_k^q+\ell)$ and B_{G^-} guarantees that every job of

2. The concept of schedulability composition has been introduced in [9].

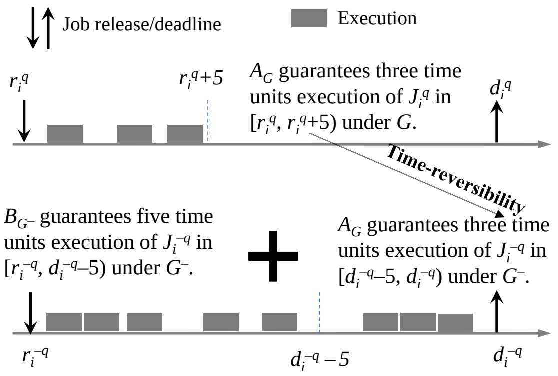

Job release/deadline
Execution
r_i^q
r_i^q+5
A_G guarantees three time units execution of J_i^q in $[r_i^q, r_i^q+5)$ under G.
d_i^q
Time-reversibility
B_{G^-} guarantees five time units execution of J_i^{-q} in $[r_i^{-q}, d_i^{-q}-5)$ under G^-.
A_G guarantees three time units execution of J_i^{-q} in $[d_i^{-q}-5, d_i^{-q})$ under G^-.
r_i^{-q}
$d_i^{-q}-5$
d_i^{-q}

Fig. 6. Schedulability composition using time-reversibility with respect to execution-level schedulability (Theorem 3)

$\tau_k \in \tau$ under G^- (denoted by J_k^{-q}) finishes its execution at least as much as $C_k - C'_k$ in $[r_k^{-q}, d_k^{-q} - \ell)$.

Proof: By Definition 2, A_G guarantees that every job of τ_i under G^- (denoted by J_i^{-q}) finishes its execution at least as much as C'_i in $[d_i^{-q} - \ell, d_i^{-q})$ (or the amount of the remaining execution at $d_i^{-q} - \ell$ if it is less than C'_i). Since B_{G^-} guarantees $C_i - C'_i$ amount of execution of J_i^{-q} in $r_i^{-q}, d_i^{-q} - \ell)$, we can guarantee that the full execution of J_i^{-q} is finished in $[r_i^{-q}, d_i^{-q})$. □

Fig. 6 illustrates an example of Theorem 3. Suppose that a time-reversible schedulability test A_G guarantees three time units execution of J_i^q in $[r_i^q, r_i^q+5)$ under G and another schedulability test B_{G^-} guarantees five time units execution of J_i^{-q} in $[r_i^{-q}, d_i^{-q} - 5)$ under G^-. Then, we can guarantee eight time units execution of J_i^{-q} in $[r_i^{-q}, d_i^{-q})$: the first five units execution by B_{G^-} and the next three units execution by a time-reversible schedulability test A_G.

One may wonder how we can find a proper C'_i that yields schedulability guarantee, efficiently. Although we do not have any optimal way, the time-complexity is not critical because of two reasons. First, since we are usually interested in offline schedulability guarantee, we can test all possible integer C'_i in $[1, C_i]$, whose time-complexity will be discussed in Section 6. Second, if time-complexity really matters, we can test only some of candidates, e.g., $C'_i \in \{0.1 \cdot C_i, 0.2 \cdot C_i, \cdots, C_i\}$, which does not compromise correctness; instead, the more candidates to be tested, the higher probability to find C'_i that yields schedulability guarantee.

While Theorem 3 has enormous potential in improving schedulability, the theorem does not exploit slack reclamation for the time-reversed scheduling algorithm. For example, if we use RTA for LRF (i.e., Lemma 2 with applying Eq. (8)) to guarantee schedulability for EDF, we cannot utilize the slack value under EDF. This is because, Eq. (8) cannot accommodate the slack value under EDF. This potentially loses the chance of deriving a tighter schedulability test by slack reclamation. To this end, we need to accommodate the slack value under a time-reversed scheduling algorithm.

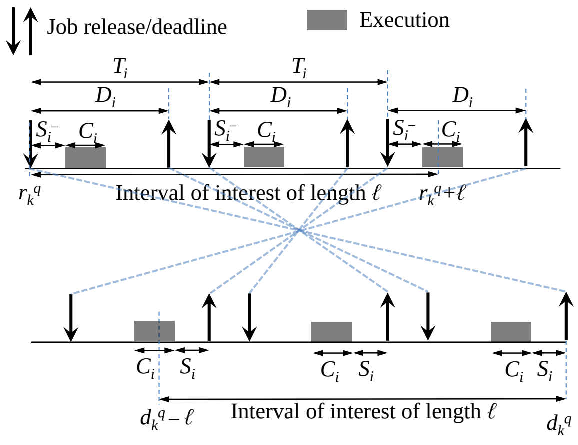


Fig. 7. An upper-bound of interference $I_{k\leftarrow i}(d_k^q - \ell, d_k^q)$ under EDF: $L_i(\ell, S_i)$

Let S_i^- denote the reversed slack value; a job J_i^q does not executes S_i^- amount of time from its release time as shown in Fig. 7 (shown in the upper figure). Then, $I_{k\leftarrow i}(r_k^q, r_k^q + \ell)$ under LRF when J_i^q does not execute S_i^- amount of time from its release, is calculated by $L_i(\ell, S_i^-)$ as follows.

$$L_i(\ell, S_i^-) = \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot C_i + \max\left(0, \min\left(C_i, \ell - \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot T_i - S_i^-\right)\right). \tag{9}$$

By definition, $L_i(\ell, S_i^-)$ can be an upper-bound of $I_{k\leftarrow i}(r_k^q, r_k^q + \ell)$ under LRF *only when* jobs of τ_i do not execute S_i^- amount of time from their release. Therefore, $L_i(\ell, S_i^-)$ cannot be an upper-bound of $I_{k\leftarrow i}(r_k^q, r_k^q + \ell)$ under vanilla LRF, because LRF does not restrict the execution from each job's release time. Instead, we can use $L_i(\ell, S_i^-)$ for EDF. That is, $L_i(\ell, S_i)$ can be an upper-bound of $I_{k\leftarrow i}(d_k^q - \ell, d_k^q)$ under EDF when S_i is the slack value of jobs of τ_i under EDF; note that the interval of interest for EDF is $[d_k^q - \ell, d_k^q)$, not $[r_k^q, r_k^q + \ell)$. This is because, $I_{k\leftarrow i}(r_k^q, r_k^q+\ell)$ under LRF with S_i^- (shown in the upper figure of Fig. 7) corresponds to $I_{k\leftarrow i}(d_k^q-\ell, d_k^q)$ under EDF with S_i (shown in the lower figure). Therefore, $I_{k\leftarrow i}(d_k^q - \ell, d_k^q)$ under EDF is upper-bounded as follows.

$$\begin{aligned} I_{k\leftarrow i}(d_k^q - \ell, d_k^q) \text{ under EDF} &\le \min\left(W_i(\ell, S_i), L_i(\ell, S_i)\right) \\ &= L_i(\ell, S_i). \end{aligned} \tag{10}$$

Applying the above inequality to Theorem 3, we can develop an EDF schedulability test as follows.

Lemma 6. A task set τ is schedulable by EDF, if for every $\tau_k \in \tau$, there exist $C'_k \in [0, C_k]$ and $\ell \in [0, D_k]$ such that the following two inequalities hold:

$$C_k - C'_k + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau - \{\tau_k\}} \min\left(W_i(\ell, S_i), E_i(D_k, S_i), \ell - (C_k - C'_k) + 1\right) \right\rfloor \le \ell, \tag{11}$$

$$C'_k + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau - \{\tau_k\}} \min\left(L_i(D_k - \ell, S_i), (D_k - \ell) - C'_k + 1\right) \right\rfloor \le D_k - \ell. \tag{12}$$

Proof: We divide the interval of interest $[r_k^q, d_k^q)$ of length D_k into two: $[r_k^q, r_k^q+\ell)$ and $[r_k^q+\ell, d_k^q)$. Then, we prove that (a) $C_k - C_k'$ amount of execution is performed in the former interval, and (b) C_k' amount of execution is performed in the latter interval.

Case (a): A job cannot execute only when there are other m jobs whose priorities are higher than the job of interest. Therefore, from Eq. (1), we guarantee $C_k - C_k'$ amount of execution performed in $[r_k^q, r_k^q+\ell)$ of length ℓ, if the following inequality holds:

$$C_k - C_k' + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau - \{\tau_k\}} I_{k\leftarrow i}(r_k^q, r_k^q+\ell), \ell - (C_k - C_k') + 1\Big) \right\rfloor \le \ell.$$

Since $I_{k\leftarrow i}(r_k^q, r_k^q+\ell) \le \min\big(W_i(\ell, S_i), E_i(D_k, S_i)\big)$ holds under EDF (from Eq. (5)), Eq. (11) implies that we can guarantee $C_k - C_k'$ amount of execution performed in $[r_k^q, r_k^q+\ell)$.

Case (b): Similar to Eq. (1), we also guarantee C_k' amount of execution performed in $[r_k^q+\ell, d_k^q)$ of length $D_k - \ell$, if the following inequality holds:

$$C_k' + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau - \{\tau_k\}} \min\Big(I_{k\leftarrow i}(r_k^q+\ell, d_k^q), (D_k - \ell) - C_k' + 1\Big) \right\rfloor.$$
$$\le D_k - \ell$$

Since $I_{k\leftarrow i}(r_k^q+\ell, d_k^q) \le L_i(D_k - \ell, S_i)$ holds under EDF (from Eq. (10)), Eq. (12) implies that we can guarantee C_k' amount of execution performed in $[r_k^q+\ell, d_k^q)$.

The lemma holds by Cases (a) and (b). □

Section 6 will demonstrate via simulation that Lemma 6 is effective in finding additional EDF-schedulable task sets. The section will also discuss time-complexity of Lemma 6.

5 Generalization of Time-Reversibility for Job-Level Dynamic-Priority Scheduling

In the previous sections, we gave formal definitions of time-reversibility and developed theories thereof for schedulability improvement. However, the definitions cannot accommodate dynamic job-parameters that vary with time such as the time to deadline and the remaining execution time at an arbitrary time instant. In this section, we generalize the definitions of time-reversibility for job-level dynamic-priority scheduling. Then, we perform cases studies—investigating time-reversibility of schedulability tests for a job-level dynamic-priority scheduling algorithm and adapting the time-reversibility theories to the tests.

5.1 Generalization of time-reversibility definitions

For job-level dynamic-priority scheduling under which a job priority may vary with time, we need to address dynamic job-parameters such as the time to deadline and the remaining execution time at an arbitrary instant. To this end, we investigate and generalize R$_1$–R$_3$ in Section 3 so as to accommodate dynamic job-parameters.

If we focus on R$_1$, it matches the job release time and deadline only. Beyond matching the simple parameters, we need to map every instant within an interval between the release time and deadline of each job as follows.

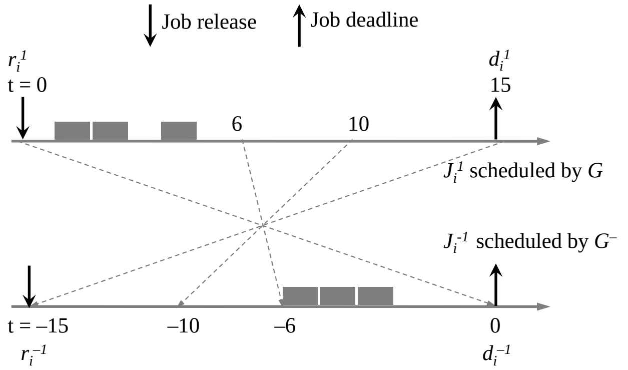


Fig. 8. Time-reversibility for job-level dynamic-priority scheduling: mapping an arbitrary instant with remaining/performed execution

R$_1'$. A time instant $-r_i^q + \alpha$ $(0 \le \alpha \le D_i)$ for J_i^{-q} is mapped to $d_i^q - \alpha$ for J_i^q.

For example, since $t = 6$ of J_i^1 in Fig. 8 is expressed by $d_i^1 - \alpha = 15 - 9$, $t = 6$ of J_i^1 is mapped to $t = -r_i^1 + \alpha = -15 + 9 = -6$ of J_i^{-1}. Similarly, $t = 10$ of J_i^1 is mapped to $t = -10$ of J_i^{-1}.

To address dynamic states of each job regarding the remaining/performed execution, R$_2'$ should be generalized as follows.

R$_2'$. The worst-case execution time of J_i^{-q} is set to that of J_i^q. And, the amount of remaining execution (*likewise* performed execution) at $-r_i^q + \alpha$ $(0 \le \alpha \le D_i)$ for J_i^{-q} is mapped to the amount of performed execution (*likewise* remaining execution) at $d_i^q - \alpha$ for J_i^q.

For example, the amount of performed execution of J_i^1 at $t = 6$ (3 units in Fig. 8) in Fig. 8 is mapped to the amount of remaining execution of J_i^{-1} at $t = -6$ (3 units in the figure).

Finally, the priority of a job should be expressed for an arbitrary instant as follows.

R$_3'$. The priority of J_i^{-q} at $-r_i^q + \alpha$ $(0 \le \alpha \le D_i)$ is set to that of J_i^q at $d_i^q - \alpha$.

Similar to Definition 1, we can define a time-reversed scheduling algorithm using R$_1'$–R$_3'$ as follows.

Definition 3. Suppose that for a given $\{J_i^q\}_{\tau_i \in \tau}$ which is prioritized by a scheduling algorithm G, $\{J_i^{-q}\}_{\tau_i \in \tau}$ is generated according to R$_1'$–R$_3'$. Then, we can derive a corresponding scheduling algorithm G^-, such that G^- directly assigns job priorities to $\{J_i^{-q}\}_{\tau_i \in \tau}$. A scheduling algorithm G^- is said to be a *time-reversed scheduling algorithm* against G.

While Definition 1 is valid only for *job-level fixed-priority* scheduling algorithms, Definition 3 can accommodate both *job-level fixed-priority* and *job-level dynamic-priority* scheduling algorithms. In order words, Definition 3 is a generalization of Definition 1 as follows. First, if we apply α to 0 and D_i, R$_1'$ is equivalent to R$_1$. Second, R$_2'$ literally subsumes R$_2$.

Finally, since the priority of a job under any job-level fixed-priority scheduling does not change over time, R_3' subsumes R_3.

Among job-level dynamic-priority scheduling algorithms, many of them (e.g., EDZL [7], RMZL [13], DMZL [13], and LLF [14]) prioritize jobs using a notion of *laxity*. The *laxity* of a job at a time instant is defined as the difference between the time to its deadline and the remaining execution of the job at the instant. By R_1' and R_2', the time to the deadline of J_i^q matches the time from the release time of J_i^{-q}, and the remaining execution at α ahead of the deadline of J_i^q maps to the performed execution at α after the release time of J_i^{-q}. Therefore, we can define *reversed-laxity* under a scheduling algorithm as opposed to laxity under its time-reversed scheduling algorithm as follows. The *reversed-laxity* of a job at a time instant is defined as the difference between the time from its release time and the performed execution at the instant. For example, while the laxity of J_i^{-1} at $t = -6$ in Fig. 8 is 6 (time to the deadline) - 3 (the remaining execution) = 3, the reversed-laxity of J_i^1 at $t = 6$ in the figure is 6 (time from the release time) - 3 (the performed execution) = 3.

Now, we present two examples of time-reversed scheduling algorithms of job-level dynamic-priority scheduling algorithms.

Example 5.1. Since J_i^q's laxity matches J_i^{-q}'s reversed-laxity, scheduling of $\{J_i^q\}_{\tau_i \in \tau}$ by the zero-laxity policy (that gives the highest priority to a job with the *zero-laxity state*) corresponds to that of $\{J_i^{-q}\}_{\tau_i \in \tau}$ by a scheduling policy that gives the highest priority to a job with the *zero-reversed-laxity state*. Therefore, EDZL (Earliest Deadline first until Zero-Laxity) that gives the highest priority to a job with the zero-laxity state and schedules other jobs by EDF corresponds to LRZRL (Latest Release-time first until Zero-Reversed-Laxity) that gives the highest priority to a job with the zero-reversed-laxity state and schedules other jobs by LRF. In other words, LRZRL is a time-reversed scheduling algorithm against EDZL (denoted by LRZRL $=$ EDZL$^-$). Similarly, EDZL $=$ LRZRL$^-$ holds.

Example 5.2. Since RM $=$ RM$^-$ holds and the zero-laxity policy matches the zero-reversed-laxity policy, scheduling of $\{J_i^q\}_{\tau_i \in \tau}$ by RMZL (Rate Monotonic until Zero Laxity) corresponds that of $\{J_i^{-q}\}_{\tau_i \in \tau}$ by the scheduling algorithm RMZRL (Rate Monotonic until Zero Reversed-Laxity). In other words, RMZL $=$ RMZRL$^-$ and RMZRL $=$ RMZL$^-$ hold. Similarly, the same relationship holds for DMZL (Deadline Monotonic until Zero Reversed-Laxity) and DMZRL (Deadline Monotonic until Zero Reversed-Laxity).

Once we find a time-reversed job-level dynamic-priority scheduling algorithm, we can apply Definition 2 to a schedulability test for a job-level dynamic-priority scheduling algorithm. In the next subsections, we investigate time-reversibility of schedulability tests for EDZL and its time-reversed scheduling algorithm LRZRL, and demonstrate how to adapt time-reversibility theories for EDZL schedulability improvement.

5.2 EDZL and LRZRL schedulability tests

EDZL (Earliest Deadline first until Zero Laxity) deploys the zero-laxity policy on top of EDF. Different from EDF, EDZL exhibits an additional necessary deadline-miss condition, which potentially makes its schedulability tighter. In this subsection, we develop RTA for EDZL that employs the necessary deadline-miss condition tailored to EDZL, and then develop RTA for LRZRL (i.e., a time-reversed scheduling algorithm against EDZL) and show its time-reversibility.

The first step to derive RTA for EDZL is to derive an upper-bound of $I_{k\leftarrow i}(r_k^q, r_k^q + \ell)$ under EDZL, which was derived from existing DA for EDZL [15]. That is, a job J_i^p can interfere with another job J_k^q only when (i) the deadline of J_i^p is no later than that of J_k^q or (ii) J_i^q has the zero laxity. The former was already addressed by $E_i(D_k, S_i)$ in Eq. (4) as EDF interference upper-bound. Even under the latter situation, a tighter upper-bound of interference is still $E_i(D_k, S_i)$ as explained in [15]. Therefore, we use the same upper-bound as EDF, i.e., Eqs. (5) and (6) for EDZL with and without slack reclamation, respectively.

Using the above upper-bounds, Lemmas 2 and 3 can guarantee the schedulability of a task set, by checking whether a task's jobs can trigger the first deadline miss. In addition, the zero-laxity-based scheduling algorithm that gives the highest priority to zero-laxity jobs has an additional necessary deadline miss condition. That is, a deadline miss occurs only when there are at most $m + 1$ jobs with the zero-laxity state under any zero-laxity-based scheduling algorithm [6]. While the condition was incorporated into DA [6, 15], it has not been into RTA. Now we develop a way to check the capability for a task to reach the zero-laxity state, to be incorporated into RTA for EDZL.

Lemma 7 (Implicitly presented in [15]). Suppose that τ is scheduled by a zero-laxity-based scheduling algorithm that gives the highest-priority to zero-laxity jobs. A task $\tau_k \in \tau$ cannot reach the zero-laxity state, if its slack value S_k is positive.

Proof: By the definition, the slack value S_k means a lower-bound of the interval between a completion time and deadline of every job invoked by τ_k. Therefore, a positive slack S_k implies that τ_k cannot reach the laxity smaller than S_k, which proves the lemma. □

Using Lemma 7, RTA for EDZL with/without slack reclamation operates as follows. Initial procedures are the same as those for RTA for EDF. After checking all tasks' schedulability by calculating their response times, we deem the task set schedulable if either all tasks are deemed schedulable (the same condition as RTA for EDF) or there are at most m tasks whose slack values are not positive by Lemma 7 (i.e., whose response times are strictly smaller than their relative deadlines).

When it comes to LRZRL, a time-reversed scheduling algorithm against EDZL, it gives the highest priority to zero-revered-laxity jobs and prioritizes other jobs by LRF. To develop RTA for LRZRL, we need to calculate an upper-bound of interference. Since a job J_i^p can interfere with another job J_k^q only when (i) the release time of J_i^p is no earlier than that of J_k^q or (ii) J_i^p has the zero-reversed-laxity. The former situation is the same as LRF, and therefore the interference is upper-bounded by $L_i(\ell)$ as presented in Eq. (8). For the latter, we need to figure out the condition for a job to have the zero-reversed-laxity. By definition, a job J_i^p has the zero reversed-laxity at t, only when it performs its execution during $[r_i^p, t)$. Therefore, if we shift the release times of jobs of τ_i earlier than the situation that yields $L_i(\ell)$ in Fig. 5, the first job should continue to perform its execution from its release time. This yields exclusion of some execution of the first job from the interference, and therefore the shift does not increase the interference. Therefore, under LRZRL, an upper-bound of $I_{k\leftarrow i}(r_k^q, r_k^q+\ell)$ is still $L_i(\ell)$.

Thus, RTA for LRZRL is the same as RTA for LRF (i.e., Lemma 2 with applying Eq. (8)). This implies that RTA for LRZRL is also time-reversible with respect to execution-, task- and task-set-level schedulability as we proved in Lemma 5.

5.3 EDZL schedulability improvement using time-reversibility

In this subsection, we show how to compose schedulability from a schedulability test for EDZL and a time-reversible schedulability test for LRZRL. In order to utilize EDZL's own necessary deadline-miss conditions related to zero-laxity tasks, we need to develop a way to compose a guarantee for every job of a task not to reach the zero-laxity state, which is different from Theorem 3 that composes a guarantee for every job of a task to finish its execution within its deadline.

To this end, we apply a simple necessary condition for a job not to reach the zero-laxity state: a job J_k^q cannot reach the zero laxity if it finishes its execution at or before d_k^q-1. That is, as long as J_k^q finishes its execution before d_k^q-1, the job's laxity at any instant in $[r_k, d_k-1]$ is at least one (because the time to deadline is always strictly larger than the remaining execution).

In order to adapt Theorem 3 so as to check each task's capability in reaching the zero-laxity state, we need to upper-bound $I_{k\leftarrow i}(d_k^q-\ell, d_k^q-1)$ under EDZL. To utilize existing results of RTA for LRZRL, we first upper-bound $I_{k\leftarrow i}(d_k^q-\ell, d_k^q)$ under EDZL. As we mentioned in Section 5.2, a job J_i^p can interfere with another job J_k^q under EDZL only when (i) the deadline of J_i^p is no later than that of J_k^q or (ii) J_i^q has the zero laxity. The interference upper-bound for Case (i) was already addressed by $L_i(\ell, S_i)$ in Eq. (10) as EDF interference upper-bound, and that for Case (ii) is also $L_i(\ell, S_i)$ in that shifting the release pattern later than Fig. 7 (shown in the lower figure) cannot increase the amount of interference. Therefore, $I_{k\leftarrow i}(d_k^q-\ell, d_k^q)$ under EDZL is upper-bounded as follows.

$$\begin{aligned} I_{k\leftarrow i}(d_k^q-\ell, d_k^q) \text{ under EDZL} &\le \min\big(W_i(\ell, S_i), L_i(\ell, S_i)\big)\\ &= L_i(\ell, S_i). \end{aligned} \tag{13}$$

Using the above inequality, $I_{k\leftarrow i}(d_k^q-\ell, d_k^q-1)$ can be upper-bounded as follows.

$$\begin{aligned} &I_{k\leftarrow i}(d_k^q-\ell, d_k^q-1) \text{ under EDZL}\\ &\le \min\big(\ell-1, I_{k\leftarrow i}(d_k^q-\ell, d_k^q) \text{ under EDZL}\big)\\ &\le \min(\ell-1, L_i(\ell, S_i)). \end{aligned} \tag{14}$$

Note that $\ell-1$ comes from the fact that $I_{k\leftarrow i}(t_0, t_1)$ under any scheduling algorithm is upper-bounded by the interval length t_1-t_0.

Incorporating Eq. (14) to the necessary deadline-miss condition for EDZL, we can develop an improved EDZL schedulability test as follows.

Lemma 8. A task set τ is schedulable by EDZL, if at least one of the two following conditions holds:

- For every $\tau_k \in \tau$, there exists $C_k' \in [0, C_k]$ and $\ell \in [0, D_k]$ such that Eqs. (11) and (12) hold; or
- For at most $|\tau|-m$ tasks $\tau_k \in \tau$, there exist $C_k' \in [0, C_k]$ and $\ell \in [0, D_k-1]$ such that Eqs. (11) and (15) hold.

$$C_k' + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau-\{\tau_k\}} \min\Big(L_i(D_k-\ell, S_i), (D_k-1-\ell)-C_k'+1\Big)\right\rfloor \le D_k-1-\ell. \tag{15}$$

Proof: Since all task sets schedulable by EDF are also schedulable by EDZL [6], the first condition holds (which is the same as Lemma 6).

The second condition addresses the necessary deadline-miss condition for EDZL: a deadline miss occurs only when there exist at least $m+1$ tasks which can reach the zero-laxity. Therefore, the remaining step is to prove that if there exist $C_k' \in [0, C_k]$ and $\ell \in [0, D_k-1]$ such that Eqs. (11) and (15) hold, τ_k cannot reach the zero-laxity state.

Then, the remaining proof is similar to that of Lemma 6, as follows. We divide the interval of interest $[r_k^q, d_k^q-1)$ of length D_k-1 into two: $[r_k^q, r_k^q+\ell)$ and $[r_k^q+\ell, d_k^q-1)$. Then, we prove that (a) C_k-C_k' amount of execution is performed in the former interval (length ℓ), and (b) C_k' amount of execution is performed in the latter interval (length $D_k-1-\ell$). Since Case (a) is the same as that of Lemma 6, here we cover Case (b) only.

Case (b): By applying Eq. (1), we can guarantee C_k' amount of execution performed in $[r_k^q+\ell, d_k^q-1)$ of length $D_k-1-\ell$, if the following inequality holds:

$$C_k' + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau-\{\tau_k\}} \min\Big(I_{k\leftarrow i}(r_k^q+\ell, d_k^q), (D_k-1-\ell)-C_k'+1\Big)\right\rfloor \le D_k-1-\ell.$$

Since $I_{k\leftarrow i}(r_k^q+\ell, d_k^q-1) \le \min(D_k-1-\ell, L_i(D_k-\ell, S_i))$ holds under EDZL (from Eq. (14)), Eq. (15) implies that we can guarantee C_k' amount of execution performed in $[r_k^q+\ell, d_k^q-1)$.

This completes the proof. □

6 Evaluation

In this section, we evaluate the schedulability tests derived from the notion of time-reversibility. For quantitative schedulability improvement, we generate a number of task sets, and check that each task set is deemed schedulable

TABLE 1
The number of constrained-deadline task sets proven schedulable by $\mathsf{RTA_{EDF}}$, $\mathsf{TR_{EDF}}$, $\mathsf{RTA_{EDZL}}$, $\mathsf{TR_{EDZL}}$, $\mathsf{RTA_{WC}}$, and $\mathsf{RTA_{LRF}}$

	The number of schedulable task sets		Ratio	The number of schedulable task sets		Ratio	The number of schedulable task sets		Ratio
m	$\mathsf{RTA_{EDF}}$	$\mathsf{TR_{EDF}}$	$\frac{\mathsf{TR_{EDF}}}{\mathsf{RTA_{EDF}}}$	$\mathsf{RTA_{EDZL}}$	$\mathsf{TR_{EDZL}}$	$\frac{\mathsf{TR_{EDZL}}}{\mathsf{RTA_{EDZL}}}$	$\mathsf{RTA_{WC}}$	$\mathsf{RTA_{LRF}}$	$\frac{\mathsf{RTA_{LRF}}}{\mathsf{RTA_{WC}}}$
2	342813	351966	102.7 %	552968	556911	100.7 %	91000	97687	107.3 %
4	197068	207454	105.3 %	417651	420494	100.7 %	43903	45909	104.6 %
8	119199	130188	109.2 %	350361	352152	100.5 %	20282	20850	102.8 %
16	74741	84891	113.6 %	317569	318600	100.3 %	9006	9176	101.9 %

TABLE 2
The number of implicit-deadline task sets proven schedulable by $\mathsf{RTA_{EDF}}$, $\mathsf{TR_{EDF}}$, $\mathsf{RTA_{EDZL}}$, $\mathsf{TR_{EDZL}}$, $\mathsf{RTA_{WC}}$, and $\mathsf{RTA_{LRF}}$

	The number of schedulable task sets		Ratio	The number of schedulable task sets		Ratio	The number of schedulable task sets		Ratio
m	$\mathsf{RTA_{EDF}}$	$\mathsf{TR_{EDF}}$	$\frac{\mathsf{TR_{EDF}}}{\mathsf{RTA_{EDF}}}$	$\mathsf{RTA_{EDZL}}$	$\mathsf{TR_{EDZL}}$	$\frac{\mathsf{TR_{EDZL}}}{\mathsf{RTA_{EDZL}}}$	$\mathsf{RTA_{WC}}$	$\mathsf{RTA_{LRF}}$	$\frac{\mathsf{RTA_{LRF}}}{\mathsf{RTA_{WC}}}$
2	469330	483458	103.0 %	613284	621851	101.4 %	176238	208941	118.6 %
4	327386	339693	103.8 %	500895	505713	101.0 %	103320	113717	110.1 %
8	238768	253407	106.1 %	445316	448277	100.7 %	58553	62076	106.0 %
16	176414	192963	109.4 %	418609	420378	100.4 %	32373	33680	104.0 %

by existing schedulability tests as well as the new ones derived in this paper. Then, we compare time-complexity of the schedulability tests.

Schedulability tests to be evaluated. This section focuses on the following six schedulability tests.

- $\mathsf{RTA_{EDF}}$: RTA for EDF with slack reclamation, which is the state-of-the-art EDF schedulability test, i.e., Lemma 2 with applying Eq. (5) in this paper,
- $\mathsf{TR_{EDF}}$: Lemma 6 developed in this paper, which is an EDF schedulability test derived from the time-reversibility theories,
- $\mathsf{RTA_{EDZL}}$: RTA for EDZL with slack reclamation, which is the state-of-the-art EDZL schedulability test, presented Section 5.2 in this paper,[3]
- $\mathsf{TR_{EDZL}}$: Lemma 8 developed in this paper, which is an EDZL schedulability test derived from the time-reversibility theories,
- $\mathsf{RTA_{WC}}$: RTA for any work-conserving scheduling algorithm with slack reclamation, which is the state-of-the-art schedulability test for any work-conserving scheduling algorithm, i.e., Lemma 2 with applying $I_{k\leftarrow i}(r_k^q, r_k^q + \ell) \leq W_i(\ell, S_i)$ in this paper, and
- $\mathsf{RTA_{LRF}}$: Lemma 2 with applying Eq. (8) developed in this paper, which is an LRF schedulability test developed in this paper. Note that $\mathsf{RTA_{LRF}}$ is the same as RTA for LRZRL, as mentioned in Section 5.2.

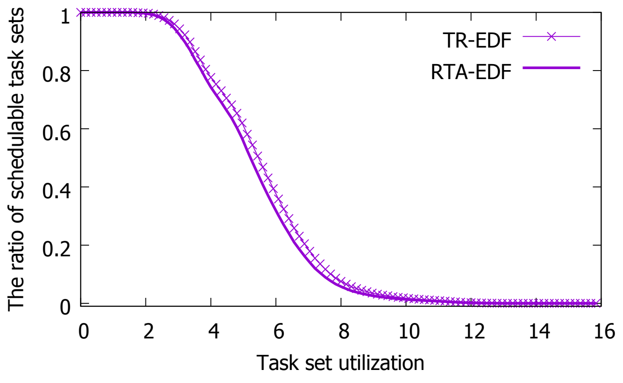
Note that it is known that all task sets deemed schedulable by DA are also deemed schedulable by the corresponding RTA [12]; for example, every task set deemed schedulable by DA for EDF with slack reclamation is also deemed schedulable by RTA for EDF with slack reclamation. Therefore, this section presents the best schedulability performance—that of RTAs, not DAs. We also note that since there was no LRF (as well as LRZRL) schedulability test so far, $\mathsf{RTA_{WC}}$ is the state-of-the-art LRF (as well as LRZRL) schedulability test.

3. Since no one developed RTA for EDZL, RTA for EDZL is our contribution. However, for fair comparison, we choose RTA for EDZL with slack reclamation as a base schedulability test, which dominates existing DA for EDZL with slack reclamation.

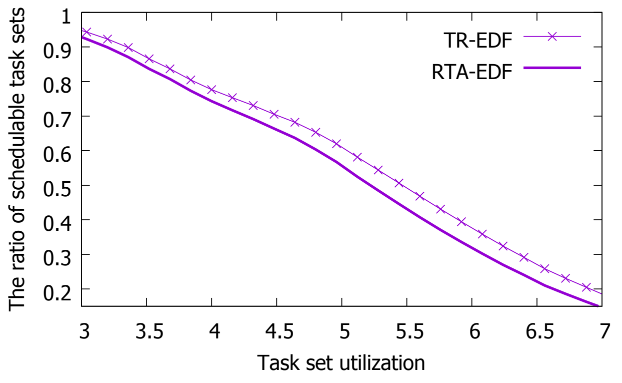
Task set generation. To demonstrate the effectiveness of time-reversibility in improving schedulability, we generate real-time task sets based on a popular technique [16], used in many multiprocessor scheduling papers such as [11, 17]. We consider three task parameters: (a) the number of processors m (2, 4, 8 or 16), (b) the type of tasks in each task set (constrained deadline: $D_i \leq T_i$ or implicit deadline: $D_i = T_i$), and (c) task utilization (C_i/T_i) distribution of individual tasks (bimodal with parameter: 0.1, 0.3, 0.5, 0.7 or 0.9, or exponential with parameter: 0.1, 0.3, 0.5, 0.7 or 0.9), detailed in [17]. For each task, T_i is uniformly chosen in $[1, 1000]$, C_i is chosen based on the bimodal or exponential parameter, and D_i is uniformly selected in $[C_i, T_i]$ for constrained-deadline tasks or is equal to T_i for implicit-deadline tasks. To meet the quantum length requirement, we set all task parameters to the closest integer values.

For each combination of (a), (b) and (c), we repeat the following steps, and generate 100,000 task sets. As a result, 1,000,000 task sets are generated, for given m (i.e., the number of processors) and the type of tasks in each task set (i.e., either implicit- or constrained-deadline task).

1) We generate a set of $m + 1$ tasks, because a task set with m or less tasks is trivially schedulable.
2) We check whether the generated task set can pass an exact feasibility condition (i.e., $\sum_{\tau\in\tau} C_i/T_i \leq m$) [18] for implicit-deadline task sets and a necessary feasibility condition in [19] for constrained-deadline ones.
3) If it fails to pass the feasibility test, we discard the generated set and return to Step 1). Otherwise, we include this set for evaluation. This valid task set serves as a basis for the next new set; we add a new task into the valid task set, and return to Step 2) with this new set.

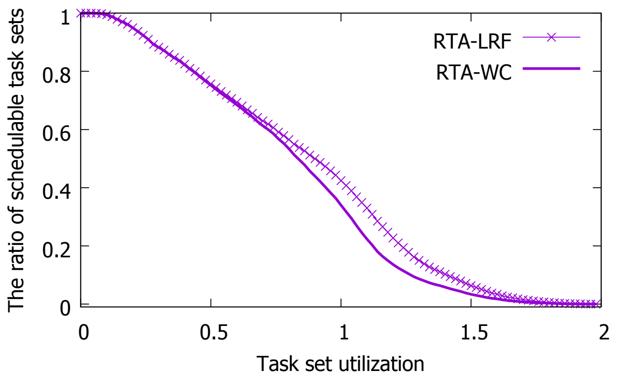


(a) All range of task set utilization

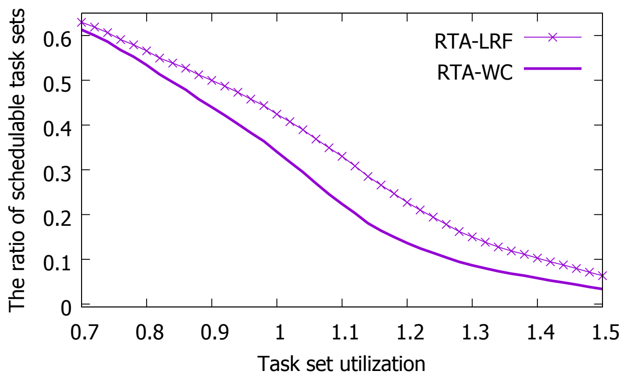


(b) Task set utilization focused on $[3.0, 7.0)$

Fig. 9. The ratio of schedulable constrained-deadline task sets by TR_{EDF} and RTA_{EDF} when $m = 16$



(a) All range of task set utilization



(b) Task set utilization focused on $[0.7, 1.5)$

Fig. 10. The ratio of schedulable implicit-deadline task sets by RTA_{LRF} and RTA_{WC} when $m = 2$

Schedulability improvement. In Tables 1 and 2, we present the number of schedulable task sets by the six schedulability tests and the ratio between the corresponding schedulability tests, on 2, 4, 8 and 16 processors. In particular, Tables 1 and 2 deal with constrained- and implicit-deadline task sets, respectively.

If we compare RTA_{EDF} with TR_{EDF}, TR_{EDF} covers up to 13.6% additional EDF-schedulability task sets, and the largest improvement is achieved for constrained-deadline task sets on $m = 16$. The improvement ratio increases as m increases, and the improvement for constrained-deadline task sets is larger than that for implicit-deadline task sets. To show the schedulability improvement according to task set utilization (i.e., $\sum_{\tau_i \in \tau} C_i/T_i$), we draw Figs. 9(a) and 9(b) for the case of constrained-deadline tasks on $m = 16$. The X-axis and Y-axis of the figures represent task set utilization and the ratio of schedulable task sets. While Fig. 9(a) illustrates all range of task set utilization, Fig. 9(b) focuses on task set utilization between 3.0 and 7.0, where the improvement is significant. As seen in the figures, the improvement is highlighted when task set utilization is between 3.0 and 7.0. This is because task sets with low (*likewise* high) utilization is inherently easy (*likewise* difficult) to schedule, yielding small room for further improvement. In the supplement, we show more graphs with different m and task type (i.e., implicit- and constrained-deadline task).

When it comes to EDZL schedulability improvement, TR_{EDZL} yields up to 1.4% schedulability improvement compared to RTA_{EDZL}. The amount of schedulability improvement for EDZL is less significant than that for EDF. This is because, RTA_{EDZL} utilizes the necessary deadline-miss condition specialized for EDZL effectively, and therefore the test is already tight enough, yielding small room for further improvement. However, the notion of time-reversibility can result in schedulability improvement even for EDZL, in spite of small quantity.

RTA_{LRF}, the first schedulability test tailored to LRF, significantly improves the state-of-the-art schedulability test, RTA_{WC}. For example, if we focus on implicit-deadline tasks on $m = 2$, there is 18.6% schedulability improvement. Similar to Figs. 9(a) and 9(b), we draw Figs. 10(a) and 10(b) for the case of constrained-deadline tasks on $m = 2$. The figures show that schedulability improvement stands out when task set utilization is in $[0.7, 1.5)$, as task sets with middle utilization have much room for further improvement, compared to those with low and high utilization.

Time-complexity. One may wonder additional time-complexity incurred by the notion of time-reversibility, but it depends on schedulability tests that the notion is applied to. Therefore, we compare time-complexity of TR_{EDF} and TR_{EDZL}, with corresponding existing schedulability tests. For time-complexity, it is known that RTA without and with slack reclamation requires $O\big(n^2 \cdot \max_{\tau_i \in \tau} D_i\big)$ and $O\big(n^3 \cdot (\max_{\tau_i \in \tau} D_i)^2\big)$ computations, respectively [4]. The former includes $\mathsf{RTA}_{\mathsf{LRF}}$, and the latter includes $\mathsf{RTA}_{\mathsf{EDF}}$, $\mathsf{RTA}_{\mathsf{EDZL}}$, and $\mathsf{RTA}_{\mathsf{WC}}$. When it comes to $\mathsf{TR}_{\mathsf{EDF}}$ and $\mathsf{TR}_{\mathsf{EDZL}}$, they requires $C_i + 1$ values to be checked for each task's C_i', yielding $O\big(n^3 \cdot (\max_{\tau_i \in \tau} D_i)^2 \cdot \max_{\tau_i \in \tau} C_i\big)$ time-complexity. Considering these schedulability tests are usually designed for offline schedulability guarantees, all the schedulability tests derived in this paper $\mathsf{RTA}_{\mathsf{LRF}}$, $\mathsf{TR}_{\mathsf{EDF}}$ and $\mathsf{TR}_{\mathsf{EDZL}}$ are practical in terms of time-complexity.

7 Conclusion

In this paper, we proposed a new paradigm for real-time scheduling, called time-reversibility, and demonstrated how to exploit the paradigm for schedulability improvement. We also showed wide applicability of time-reversibility; it can be applied to not only simple scheduling algorithms such as EDF, but also job-level dynamic-priority scheduling algorithms such as EDZL.

While the target system model was limited to preemptive scheduling algorithms and sequential tasks, we believe that the notion of time-reversibility can be applied to more general system models. In the future, we would like to study how to adapt time-reversibility for other system models, such as non-preemptive scheduling algorithm, parallel tasks [20], mixed-criticality tasks [21], and end-to-end periodic tasks [22].

Acknowledgement

An earlier (shorter) version of this paper was presented at the IEEE RTSS 2014 [23]. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014R1A1A1035827).

References

[1] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[2] T. P. Baker and M. Cirinei, "Brute-force determination of multiprocessor schedulability for sets of sporadic hard-deadline tasks," in *Proceedings of the 11th International Conference on Principles of Distributed Systems*, 2007, pp. 62–75.

[3] V. Bonifaci and A. Marchetti-Spaccamela, "Feasibility analysis of sporadic real-time multiprocessor task systems," *Algorithmica*, vol. 63, no. 4, pp. 763–780, 2012.

[4] M. Bertogna and M. Cirinei, "Response-time analysis for globally scheduled symmetric multiprocessor platforms," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2007, pp. 149–160.

[5] M. Bertogna, M. Cirinei, and G. Lipari, "Improved schedulability analysis of EDF on multiprocessor platforms," in *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 2005, pp. 209–218.

[6] J. Lee, A. Easwaran, I. Shin, and I. Lee, "Zero-laxity based real-time multiprocessor scheduling," *Journal of Systems and Software*, vol. 84, no. 12, pp. 2324–2333, 2011.

[7] S. Cho, S.-K. Lee, S. Ahn, and K.-J. Lin, "Efficient real-time scheduling algorithms for multiprocessor systems," *IEICE Trans. on Communications*, vol. E85–B, no. 12, pp. 2859–2867, 2002.

[8] A. Mok, "Fundamental design problems of distributed systems for the hard-real-time environment," Ph.D. dissertation, Massachusetts Institute of Technology, 1983.

[9] J. Lee, K. G. Shin, I. Shin, and A. Easwaran, "Composition of schedulability analyses for real-time multiprocessor systems," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 941–954, 2015.

[10] J. Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic real-time tasks," *Performance Evaluation*, vol. 2, pp. 237–250, 1982.

[11] M. Bertogna, M. Cirinei, and G. Lipari, "Schedulability analysis of global scheduling algorithms on multiprocessor platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, pp. 553–566, 2009.

[12] M. Bertogna and S. Baruah, "Tests for global EDF schedulability analysis," *Journal of systems architecture*, vol. 57, no. 5, pp. 487–497, 2011.

[13] R. I. Davis and A. Burns, "FPZL schedulability analysis," in *Proceedings of IEEE Real-Time Technology and Applications Symposium (RTAS)*, 2011, pp. 245–256.

[14] J. Y.-T. Leung, "A new algorithm for scheduling periodic, real-time tasks," *Algorithmica*, vol. 4, pp. 209–219, 1989.

[15] T. P. Baker, M. Cirinei, and M. Bertogna, "EDZL scheduling analysis," *Real-Time Systems*, vol. 40, pp. 264–289, 2008.

[16] T. P. Baker, "Comparison of empirical success rates of global vs. partitioned fixed-priority EDF scheduling for hard real-time," Department of Computer Science, Florida State University, Tallahassee, Tech. Rep. TR–050601, 2005.

[17] J. Lee, A. Easwaran, and I. Shin, "Laxity dynamics and LLF schedulability analysis on multiprocessor platforms," *Real-Time Systems*, vol. 48, no. 6, pp. 716–749, 2012.

[18] S. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, "Proportionate progress: a notion of fairness in resource allocation," *Algorithmica*, vol. 15, no. 6, pp. 600–625, 1996.

[19] T. P. Baker and M. Cirinei, "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)*, 2006, pp. 178–190.

[20] H. S. Chwa, J. Lee, K.-M. Phan, A. Easwaran, and I. Shin, "Global EDF schedulability analysis for synchronous parallel tasks on multicore platforms," in *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 2013, pp. 25–34.

[21] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2007, pp. 239–243.

[22] J. Yao, X. Liu, Z. Gu, X. Wang, and J. Li, "Online adaptive utilization control for real-time embedded multiprocessor systems," *Journal of Systems Architecture*, vol. 56, no. 9, pp. 463–473, 2010.

[23] J. Lee, "Time-reversibility of schedulability tests," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2014, pp. 294–303.

Jinkyu Lee is an assistant professor in Department of Computer Science and Engineering, Sungkyunkwan University (SKKU), Republic of Korea, where he joined in 2014. He received the B.S., M.S., and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea, in 2004, 2006, and 2011, respectively. He has been a visiting scholar/research fellow in the Department of Electrical Engineering and Computer Science, University of Michigan, U.S.A. in 2011–2014. His research interests include system design and analysis with timing guarantees, QoS support, and resource management in real-time embedded systems, mobile systems, and cyber-physical systems. He won the best student paper award from the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) in 2011, and the Best Paper Award from the 33rd IEEE Real-Time Systems Symposium (RTSS) in 2012.