Response Time Analysis for Real-Time Global Gang Scheduling

Seongtae Lee, Seunghoon Lee, and Jinkyu Lee†

Department of Computer Science and Engineering
Sungkyunkwan University (SKKU), Republic of Korea
{yuns0509, seunghoon.l, jinkyu.lee}@skku.edu

2022 IEEE Real-Time Systems Symposium (RTSS) | 978-1-6654-5346-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/RTSS55097.2022.00018

***Abstract*—This paper aims at developing a tight schedulability analysis for real-time global gang scheduling, in which threads of each task subject to timing requirements are assigned to multiple processors in parallel (i.e., following the rigid gang task model). Focusing on the RTA (Response Time Analysis) framework known to exhibit high schedulability performance for other task models, we address two following issues: i) how to generalize the existing RTA framework to gang scheduling and utilize existing RTA components of other task models for the generalized framework, and ii) how to incorporate important characteristics of gang scheduling into the RTA framework in a systematic way to minimize the framework's pessimism in judging schedulability. By addressing the issues, our RTA framework enables to derive tight schedulability analysis for EDF, FP and potentially more scheduling algorithms for real-time global gang scheduling. Also, our simulation results demonstrate that the proposed RTA framework outperforms/complements existing studies for real-time global/non-global gang scheduling, in terms of schedulability performance.**

I. Introduction

Parallel embedded architectures such as graphics processing units, due to the advantage of handling large streams of data at once, are chosen to take reciprocal benefit from the advancement of various deep learning algorithms and the industry they have made. Although the interest of parallel embedded architectures and the boost of performance in the recent decade are at their peak [1], [2], [3], comparatively little is considered on building their time-predictable and safety-critical. To support real-time tasks subject to timing constraints by utilizing parallel embedded architectures, recent studies have paid attention to real-time gang scheduling, in which threads of each real-time task are assigned to multiple processors in parallel [4], [5], [6], [7], [8], [9], [10], [11]. However, only a few of them are capable of providing timing guarantees (i.e., schedulability) of a given set of real-time tasks [5], [10], [11], and their schedulability analysis techniques have not matured yet, compared to those for real-time sequential scheduling.

Response Time Analysis (RTA) is one of the most popular schedulability analysis frameworks due to its high schedulability performance and its applicability to most (if not all) task models and scheduling algorithms. Focusing on RTA, this paper aims at addressing the following issues for real-time global[1] gang scheduling.

Q1. How to generalize the existing RTA framework to the gang task model, and how to utilize existing RTA components of other task models for the generalized framework?

Q2. How to incorporate important characteristics of gang scheduling into the RTA framework in a systematic way to minimize the framework's pessimism in judging schedulability?

To address Q1, we interpret the existing RTA framework for the sequential task model, and define some notions specialized for gang scheduling. Using the notions, we generalize the RTA framework to gang scheduling. We show that a typical way to utilize the proposed RTA framework is to separate the duration (i.e., one-dimensional value) of *interference* of a task τ_i on another task τ_k, from calculating the amount (i.e., two-dimensional value, which is the duration multiplied by the number of occupied processors) of *interference* of τ_i on τ_k; note that *interference* of τ_i on τ_k implies the state in which τ_k cannot execute due to the execution of τ_i. This separation, implicitly used in gang scheduling studies (e.g., [4], [10]), enables reusing the existing techniques to calculate upper-bounds of the duration of interference for the sequential task model, yielding a natural, yet effective generalization of the RTA framework for gang scheduling. However, the separation incurs pessimism of calculating the amount of interference due to the failure of full consideration of parallel execution behavior of gang scheduling.

As to Q2, we develop two important techniques that reduce the pessimism of the proposed RTA framework with the separation. First, we focus on the most important characteristic of gang scheduling: a task τ_k occupies m_k *multiple processors* whenever it is executed. Therefore, if the sum of m_k for a set of some tasks is larger than the number of processors, it is impossible for the tasks to execute at the same time. Since the RTA framework with the separation cannot address such *non-parallel execution constraints*, we develop a technique of i) how to efficiently find such a task set in which the sum of each task's m_k is larger than the number of processors and ii) how to calculate tight upper-bounds of the amount of interference of such a task set on a target task. Second, observing

†Jinkyu Lee is the corresponding author.

[1] In global scheduling, there is no restriction for task-processor mapping.

2576-3172/22/$31.00 ©2022 IEEE
DOI 10.1109/RTSS55097.2022.00018

over-estimation of the contribution of a task to interference in the RTA framework, we develop a systematic technique to find the amount of execution of each task that cannot contribute to interference on another task. Finally, we develop how to compose the first and second techniques. We would like to emphasize that, all techniques improve the proposed RTA framework in terms of schedulability performance, and they can be applied to any scheduling algorithm as long as its upper-bounds for the duration of interference have been derived.

To demonstrate the effectiveness of the RTA framework with the proposed techniques, we apply it to FP (Fixed-Priority) and EDF (Earliest Deadline First) [12]. Our simulation results show three important characteristics of the proposed RTA framework in terms of schedulability performance. First, if we focus on global gang scheduling with FP and EDF, the proposed RTA outperforms/complements the corresponding existing studies. Second, if we focus on any gang scheduling, the proposed RTA is superior to the existing highest-performance schedulability test for non-global gang scheduling in some settings (but not all settings). Third, the proposed two techniques and its composition significantly improve the schedulability performance of our basic RTA that addressed Q1 only.

In summary, this paper makes the following contributions.

- Generalization of the existing RTA framework for gang scheduling, and its utilization of existing RTA components of other task models,
- Development of two novel techniques and their composition to improve the RTA framework, based on the identification of two issues thereof, and
- Demonstration of the effectiveness of the proposed framework via simulation.

The rest of this paper is structured as follows. Section II explains our system model and related work. Section III generalizes the RTA framework to gang scheduling. Sections IV and V develop the two techniques with their composition to improve the RTA framework. Section VI evaluates the RTA framework, and Section VII concludes the paper.

II. Background

In this paper, we consider the *sporadic* gang task model [4], which is a generalization of the sporadic sequential task model [13]. A gang task τ_i in a task set τ is specified as (T_i, D_i, C_i, m_i), where T_i is the minimum separation, D_i $(\leq T_i)$ is the relative deadline, C_i is the worst-case execution duration, and m_i is the number of threads of τ_i to be concurrently executed (i.e., subject to the rigid gang task model [14]). A gang task τ_i invokes a series of gang jobs, and the release times of any two consecutive gang jobs of τ_i are separated by at least T_i time units. Once a job of τ_i is released at t_0, it should finish its execution until $t_0 + D_i$. The duration of the execution of a job of τ_i is at most C_i, and the job occupies exactly m_i processors whenever it is executed. A job is said to be *active* at t, if it is released no later than t but has remaining execution at t.

We consider a system equipped with m identical processors, meaning that at most m threads of gang jobs can be concurrently executed in each time slot. In this paper, we consider *global* preemptive gang scheduling; the execution of any lower-priority gang job can be preempted by that of a higher-priority gang job, and any gang job can be executed in any processor. Similar to existing studies on gang scheduling, we consider work-conserving scheduling; there are m' idle processors, only if there is no active job of τ_i with $m_i \leq m'$.

Let one time unit denote a quantum length, and a time slot denote an interval of length 1; note that all theories in this paper can be immediately applicable to a continuous time model, by replacing 1 with $\epsilon \to 0$. Let $|\tau|$ denote the number of tasks in τ. Let LHS and RHS denote the left-hand-side and the right-hand-side, respectively.

A task set τ is said to be *schedulable* by a target scheduling algorithm on an m-processor platform, if the following condition holds for every job set invoked by tasks in τ: there is no job deadline miss when a job set is scheduled by the scheduling algorithm on the m-processor platform.

Regarding global preemptive gang scheduling, most studies have focused on EDF [4], [5], [15] and FP [6], [7], while some studies have aimed to find optimal scheduling [8], [9]. Among the studies, only a few of them [4], [5], [15] have presented non-trivial schedulability analysis subject to our task model, but it is not straightforward how to apply other scheduling algorithms than EDF to the schedulability analysis; note that the schedulability analysis in the study [4] has been shown to be flawed [16]. Also, a study in [10] has focused on a new task model of bundle parallel tasks; its schedulability analysis with one bundle case can be applied to gang scheduling with FP.

When it comes to non-global preemptive gang scheduling, a study in [11] has introduced a new scheduling category called stationary scheduling (a generalization of partitioned scheduling for the sequential model) and developed its schedulability analysis for FP. Also, there exist other studies on gang scheduling subject to different system models, e.g., non-preemptive scheduling [17], [18], [19], DAG gang tasks [20], the mixed-criticality task model [21], and the soft real-time task model [22], which are out of scope for this paper.

III. Generalizing RTA Framework to Gang Scheduling and Its Utilization

In this section, we generalize the existing RTA framework for the sequential task model, to gang scheduling. To this end, we first interpret the existing RTA framework for the sequential task model for a multiprocessor platform (which is equivalent to the gang task model with $m_i = 1$ for every task $\tau_i \in \tau$) in [23]. Consider a continuous interval of length L $(\leq D_k)$ that starts at the release time of a job of τ_k of interest, called the target interval of length L for τ_k.

Definition 1: Let X denote the duration in the target interval of length L for τ_k, in which a job of τ_k is active but cannot be executed. If X is larger than $(L - C_k + 1)$, we arbitrarily select

$(L-C_k+1)$ time slots among the X time slots; otherwise, we select all X time slots. We define the selected time slots, as *k-interference time slots* of the target interval of length L for τ_k.

Then, the existence of $(L-C_k+1)$ k-interference time slots in the target interval is a necessary condition for the job of τ_k of interest not to finish its execution for C_k in the target interval. To express the contribution of jobs of τ_i to k-interference time slots, we define the following notion.

Definition 2: $I_{k\leftarrow i}(L)$ is defined as the *duration* of execution of jobs of τ_i in k-interference time slots of the target interval of length L for τ_k. We call $I_{k\leftarrow i}(L)$ the duration of interference of τ_i on τ_k in an interval of length L.

Under the sequential task model, there should be m other jobs (than a job of τ_k) executed in each k-interference slot. Therefore, the existence of $(L-C_k+1)$ k-interference time slots of the target interval implies $\sum_{\tau_i\in\tau,\tau_k\neq\tau_i} I_{k\leftarrow i}(L) = m\cdot(L-C_k+1)$. Therefore, if there exists $C_k \leq L \leq D_k$ that satisfies Eq. (1) (which is equivalent to Eq. (2) by considering the quantum length of 1), the job of τ_k finishes its execution for C_k in the target interval of length L for τ_k.

$$\sum_{\tau_i\in\tau,\tau_i\neq\tau_k} I_{k\leftarrow i}(L) < m\cdot(L-C_k+1) \tag{1}$$

$$\Longleftrightarrow C_k + \left\lfloor \frac{\sum_{\tau_i\in\tau,\tau_i\neq\tau_k} I_{k\leftarrow i}(L)}{m} \right\rfloor \leq L \tag{2}$$

To utilize Eq. (1) for an offline schedulability test, an upper-bound of $I_{k\leftarrow i}(L)$ for all jobs of τ_k (denoted by $I^{+}_{k\leftarrow i}(L)$) under each target scheduling algorithm has been derived. Then, if every $\tau_k \in \tau$ has L ($\leq D_k$) that satisfies Eq. (1) with $I_{k\leftarrow i}(L) = I^{+}_{k\leftarrow i}(L)$, τ is schedulable by the target scheduling algorithm on an m-processor platform under the sequential task model [23].

We now generalize Eq. (1) to the gang task model. For gang scheduling, we express the amount of execution, as a two-dimensional value, which is the duration (or the number of time slots) of interest multiplied by the number of processors of interest. While we keep the definitions of the target interval of length L for τ_k, k-interference slots, $I_{k\leftarrow i}(L)$ and $I^{+}_{k\leftarrow i}(L)$, we need additional notions.

Definition 3: In each k-interference time slot, there should be at least $(m-m_k+1)$ processors occupied by other jobs than a job of τ_k; otherwise, the job of τ_k executes on m_k processors, which contradicts the definition of the k-interference time slot. We arbitrarily select $(m-m_k+1)$ processors occupied by other jobs (than the job of τ_k) in a k-interference time slot. We define the selected processors, as *k-interference processors* of a k-interference time slot.

In order to extend the notion of the duration of interference (i.e., $I_{k\leftarrow i}(L)$ as one-dimensional value) to that of the amount of interference (as two-dimensional value), we define the following notion.

Definition 4: $A_{k\leftarrow i}(L)$ is defined as the *amount* of execution of jobs of τ_i on k-interference processors in k-interference time slots of the target interval of length L for τ_k. We call $A_{k\leftarrow i}(L)$ the amount of interference of τ_i on τ_k in an interval of length L.

Then, the following lemma records an exact condition for a job not to finish its execution.

Lemma 1: Eq. (3) is a necessary and sufficient condition for a job of τ_k not to finish its execution for C_k within the interval of length L ($\leq D_k$) that starts at its release time,

$$\sum_{\tau_i\in\tau,\tau_i\neq\tau_k} A_{k\leftarrow i}(L) = (m-m_k+1)\cdot(L-C_k+1) \tag{3}$$

Proof: Suppose that Eq. (3) holds. By definition, there are exactly $(m-m_k+1)$ k-interference processors in each k-interference time slot. By dividing the RHS by $(m-m_k+1)$, we have $(L-C_k+1)$ k-interference time slots, implying the job of τ_k cannot finish its execution for C_k. Therefore, sufficiency holds.

Suppose that the job of τ_k cannot finish its execution for C_k within the interval of length L, meaning that it cannot execute for at least $(L-C_k+1)$ time slots in the interval. By the definition of k-interference time slots, the job of τ_k has $(L-C_k+1)$ k-interference time slots. By the definition of k-interference processors, each k-interference time slot has exactly $(m-m_k+1)$ processors. Therefore, Eq. (3) holds. ■

Similar to the relationship between $I_{k\leftarrow i}(L)$ and $I^{+}_{k\leftarrow i}(L)$ (i.e., an upper-bound of $I_{k\leftarrow i}(L)$), let $A^{+}_{k\leftarrow i}(L)$ denote an upper-bound of $A_{k\leftarrow i}(L)$ under each target scheduling algorithm. By negating Lemma 1, we can develop the RTA framework for gang scheduling.

Theorem 1: If there exists $C_k \leq L \leq D_k$ that satisfies Eq. (4), the job of τ_k finishes its execution for C_k in the target interval of length L for τ_k.

$$\sum_{\tau_i\in\tau,\tau_i\neq\tau_k} A_{k\leftarrow i}(L) < (m-m_k+1)\cdot(L-C_k+1) \tag{4}$$

$$\Longleftrightarrow \quad C_k + \left\lfloor \frac{\sum_{\tau_i\in\tau,\tau_i\neq\tau_k} A_{k\leftarrow i}(L)}{m-m_k+1} \right\rfloor \leq L \tag{5}$$

Also, suppose that we derive $A^{+}_{k\leftarrow i}(L)$, an upper-bound of $A_{k\leftarrow i}(L)$ for all jobs of τ_k under a target scheduling algorithm, for every pair of τ_k and τ_i ($\neq \tau_k$). If every $\tau_k \in \tau$ has $C_k \leq L \leq D_k$ that satisfies Eq. (4) with $A_{k\leftarrow i}(L) = A^{+}_{k\leftarrow i}(L)$, τ is schedulable by the target scheduling algorithm on an m-processor platform. We call the minimum of such L for τ_k the response time of τ_k (denoted by R_k).

Proof: Suppose that even though Eq. (4) holds for $C_k \leq L \leq D_k$, the job of τ_k cannot finish its execution for C_k within the interval of length L starting from its release time. By the definition of $A_{k\leftarrow i}(L)$, the LHS of Eq. (4) is an upper-bound of the amount of execution of all jobs other than the job of τ_k on k-interference processors in k-interference slots. Therefore, the supposition contradicts Lemma 1, which proves the first

Algorithm 1 RTA for a set of gang tasks τ for given $\{A^{+}_{k\leftarrow i}(L)\}$

1: $S_k \leftarrow 0$ for every task $\tau_k \in \tau$
2: **while** TRUE **do**
3:   **for** every task $\tau_k \in \tau$ **do**
4:     $R_k \leftarrow C_k$
5:     **while** $R_k \leq D_k$ **do**
6:       $A^{\text{total}}_k(R_k) \leftarrow$ the numerator of the LHS of Eq. (5), by applying $A_{k\leftarrow i}(L) = A^{+}_{k\leftarrow i}(R_k)$ for $\tau_i \in \tau, \tau_i \neq \tau_k$
7:       **if** Eq. (5) holds with $L = R_k$ and $\sum A_{k\leftarrow i}(L) = A^{\text{total}}_k(R_k)$ **then**
8:         $S_k \leftarrow D_k - R_k$ and exit the inner while loop.
9:       **else**
10:         $R_k \leftarrow$ the LHS of Eq. (5) with $\sum A_{k\leftarrow i}(L) = A^{\text{total}}_k(R_k)$
11:       **end if**
12:     **end while**
13:   **end for**
14:   **if** $R_k \leq D_k$ holds for every $\tau_k \in \tau$ **then** Return schedulable
15:   **if** There is no update of any S_k for $\tau_k \in \tau$ in the current iteration of the outer while loop **then** Return unschedulable
16: **end while**

part of the theorem. Note that Eq. (4) and (5) are equivalent by applying the quantum length of 1.

Since $A_{k\leftarrow i}(L) \leq A^{+}_{k\leftarrow i}(L)$ holds by definition, the second part of the theorem also holds. ■

Using Theorem 1, Algorithm 1 calculates the response time (R_k) of every task $\tau_k \in \tau$, which is the same as that for the sequential model [23] except for the difference between the fraction part of Eqs. (2) and (5). The algorithm utilizes the slack of τ_k, denoted by S_k; a higher value of S_k will result in a smaller upper-bound of interference to be derived for a target scheduling algorithm, e.g., $W_i(L)$ and $E_{k\leftarrow i}$ in Eqs. (7) and (8). $S_k > 0$ implies RTA *guarantees* that every job of τ_k finishes S_k ahead of its deadline. On the other hand, $S_k = 0$ implies we cannot use the slack value because RTA cannot guarantee any $S_k > 0$.

From zero slack for every task (Line 1), Lines 3–13 calculate every R_k starting from $R_k = C_k$ (Line 4) as long as R_k does not exceed D_k (Line 5). We calculate $\sum_{\tau_i \in \tau, \tau_i \neq \tau_k} A^{+}_{k\leftarrow i}(R_k)$ (Line 6). If Eq. (5) holds for current $L = R_k$, update the slack value by setting S_k to $(D_k - R_k)$ and exit the inner while loop (Lines 7–8); otherwise, increase R_k based on Eq. (5) (Lines 9–10). After each iteration of Lines 3–13, if $R_k \leq D_k$ holds for every $\tau_k \in \tau$, return schedulable (Line 14). If there is no slack update, return unschedulable (Line 15). Otherwise, we repeat Lines 3–13 with new slack values.

In order to derive $A^{+}_{k\leftarrow i}(L)$, we can decompose $A_{k\leftarrow i}(L)$ (i.e., the amount of interference) into $I_{k\leftarrow i}(L)$ (i.e., the duration of interference) and the number of processors of interest.

Lemma 2 (Implicitly used in many studies, e.g., [4], [10]): Eq. (6) holds for a job of τ_k in its target interval of length L, where $\tau_i \in \tau$ ($\tau_i \neq \tau_k$).

$$A_{k\leftarrow i}(L) \leq I_{k\leftarrow i}(L) \cdot \min(m_i, m - m_k + 1) \tag{6}$$

Proof: By Definitions 1 and 3 for k-interference slots and k-interference processors, any job of τ_i cannot be executed on more than $(m - m_k + 1)$ k-interference processors. Considering any job of τ_i in a time slot is executed on m_i processors, the amount of execution of jobs of τ_i that belongs to $A_{k\leftarrow i}(L)$ cannot be larger than $I_{k\leftarrow i}(L) \cdot \min(m_i, m - m_k + 1)$, which proves the lemma. ■

Once we apply Lemma 2, we can reuse existing upper-bounds for $I_{k\leftarrow i}(L)$ (denoted by $I^{+}_{k\leftarrow i}(L)$) developed for the sequential task model as they are. For example, we can use $I^{+}_{k\leftarrow i}(L)$ under FP, derived in [23]:

$$\begin{aligned} &I^{+}_{k\leftarrow i}(L) = 0, \quad \text{if } \tau_k \text{ has a higher priority than } \tau_i, \\ &I^{+}_{k\leftarrow i}(L) \leq \min\big(W_i(L), L - C_k + 1\big), \quad \text{otherwise.} \end{aligned} \tag{7}$$

The physical meaning of $W_i(L)$ is the maximum duration of execution of jobs of τ_i in an interval of length L, and therefore $W_i(L)$ is an upper-bound of $I_{k\leftarrow i}(L)$ for any scheduling algorithm. Also, it is calculated by $W_i(L) = N_i(L) \cdot C_i + \min\big(C_i, L + D_i - S_i - C_i - N_i(L) \cdot T_i\big)$, where $N_i(L) = \big\lfloor \frac{L + D_i - S_i - C_i}{T_i} \big\rfloor$ [23]. Also, by the definitions of k-interference time slots and $I_{k\leftarrow i}(L)$, any $I^{+}_{k\leftarrow i}(L)$ cannot be larger than $(L - C_k + 1)$.

Similarly, we can use $I^{+}_{k\leftarrow i}(L)$ under EDF, derived in [23]:

$$I^{+}_{k\leftarrow i}(L) \leq \min\big(W_i(L), E_{k\leftarrow i}, L - C_k + 1\big). \tag{8}$$

The physical meaning of $E_{k\leftarrow i}$ is the maximum duration of execution of jobs of τ_i in an interval of length D_k, whose deadlines are no later than the job of τ_k of interest. $E_{k\leftarrow i}$ is calculated by $\big\lfloor \frac{D_k}{T_i} \big\rfloor \cdot C_i + \min\big(C_i, \max(0, D_k - \big\lfloor \frac{D_k}{T_i} \big\rfloor \cdot T_i - S_i)\big)$ [23].

The upper bounds of the duration of interference $I^{+}_{k\leftarrow i}(L)$ under FP and EDF in Eqs. (7) and (8) developed for the sequential task model can be applied to our RTA framework for gang scheduling in Algorithm 1 with $A^{+}_{k\leftarrow i}(L) = I^{+}_{k\leftarrow i}(L) \cdot \min(m_i, m - m_k + 1)$. This is because the upper-bounds depend only on the sequential task parameters T_i, D_i and C_i and the interval length L ($\leq D_k$), and they are not changed by the difference between sequential (i.e., $m_i = 1$) and gang tasks. Similarly, we can reuse $I^{+}_{k\leftarrow i}(L)$ for most (if not all) prioritization policies (such as EDZL, FPZL and LLF) for the sequential task model. We record the RTA for EDF and FP in the following theorem.

Theorem 2: Algorithm 1 by applying $A^{+}_{k\leftarrow i}(L) = I^{+}_{k\leftarrow i}(L) \cdot \min(m_i, m - m_k + 1)$ to Line 6 yields the schedulability analysis for FP and EDF, if we apply the RHS of Eq. (7) and that of Eq. (8) for $I^{+}_{k\leftarrow i}(L)$, respectively.

Proof: By Algorithm 1, Lemma 2, and Eqs. (7) and (8), the theorem holds. ■

As demonstrated, reusing $I^{+}_{k\leftarrow i}(L)$ developed for the sequential task model offers simplicity and applicability, without developing $A^{+}_{k\leftarrow i}(L)$ for gang scheduling under target scheduling algorithms. However, since the advantage comes from the separation between the duration of interest and the processors of interest to be executed, this incurs pessimism of calculating the amount of interference due to the failure of full consideration of parallel execution behavior of gang scheduling. Therefore, we will develop two important techniques that reduce the pessimism of the proposed RTA framework while allowing reuse of existing $I^+_{k\leftarrow i}(L)$. First, we investigate the behavior of gang scheduling and address *non-parallel execution constraints* in which some jobs cannot be executed in parallel (in Section IV). Second, we investigate how jobs of each task occupy k-interference processors, and address *over-estimation of k-interference processor occupation* (in Section V).

IV. Addressing Non-Parallel Execution Constraints

In the previous section, we succeeded to naturally generalize the existing RTA framework and enabled to utilize $I^+_{k\leftarrow i}(L)$ developed for the sequential task model. As a result, Algorithm 1 can be used as a schedulability analysis for popular prioritization policies whose $I^+_{k\leftarrow i}(L)$ for the sequential model has been developed, e.g., EDF and FP. Despite such a big advantage, Algorithm 1 with reuse of $I^+_{k\leftarrow i}(L)$ (e.g., Theorem 2) cannot fully take account of the most important characteristic of gang scheduling. That is, while any m tasks affiliated with the sequential task model can be concurrently executed on an m-processor platform, the same cannot be said to m tasks affiliated with the gang task model. For example, it is easily observed that a job of τ_i and a job of τ_k cannot be concurrently executed on an m-processor platform, if $m_i + m_k > m$.

Algorithm 1 with reuse of $I^+_{k\leftarrow i}(L)$ (e.g., Theorem 2) cannot accommodate such constraints regarding concurrent execution, called *non-parallel execution constraints*, which yields pessimistic calculation of the amount of interference, as shown in the following example.

Example 1: Three gang tasks are executed on a platform of $m = 10$ processors: $\tau_1(T_1{=}10, D_1{=}10, C_1{=}5, m_1{=}6)$, $\tau_2(10, 10, 5, 5)$, $\tau_3(5, 5, 1, 2)$. We apply FP scheduling where the priority of τ_1 and that of τ_3 are the highest and lowest, respectively. First, we can confirm τ_1 and τ_2 are schedulable for any job arrival pattern invoked by the tasks (such as two patterns in Figure 1), because their T_i and D_i are 10 while their C_i is half of 10. Second, since $m_1{+}m_2{=}6{+}5{=}11 > m{=}10$ holds, any job of τ_1 and that of τ_2 cannot be concurrently executed. Therefore, at least four processors are not occupied by τ_1 and τ_2 at any time, as shown in Figure 1, implying τ_3 cannot miss its job deadlines as well.

Apart from actual results, if we apply Theorem 2 for τ_3, the interference term for τ_1 ($I^+_{3\leftarrow 1}(L)$) *and* that for τ_2 ($I^+_{3\leftarrow 2}(L)$) are added to the numerator of the fraction in Eq. (5), yielding no guarantee of the schedulability of τ_3. This is because Theorem 1 counts $I^+_{3\leftarrow 1}(L)$ and $I^+_{3\leftarrow 2}(L)$ independently using Eq. (7), and therefore it does not take the fact that the interval which $I^+_{3\leftarrow 1}(L)$ targets should not overlap with the interval which $I^+_{3\leftarrow 2}(L)$ targets into account. That is, since

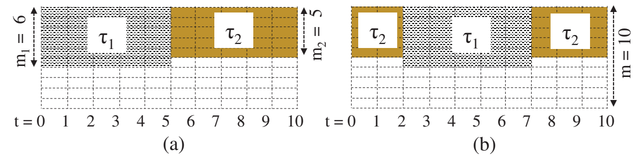


Fig. 1. Schedules of τ_1 and τ_2 in Example 1 by FP on 10 processors: (a) when a job of τ_1 and that of τ_2 are released at $t{=}0$, and (b) when a job of τ_2 is released at $t = 0$ but a job of τ_1 is released at $t{=}2$

$W_1(L) = W_2(L) = L$ holds for every $C_3{=}2 \le L \le D_3{=}5$ by Eq. (7), the numerator of the fraction in Eq. (5) for τ_3 is $L \cdot \min(m_1, m - m_3 + 1) + L \cdot \min(m_2, m - m_3 + 1) = L \cdot 6 + L \cdot 5 = L \cdot 11$. Then, Eq. (5) for τ_3 is calculated by $C_3 + \left\lfloor \frac{L\cdot 11}{m-m_3+1} \right\rfloor = 1 + \left\lfloor \frac{L\cdot 11}{9} \right\rfloor \ge 1 + L$. This means there is no $C_3 \le L \le D_3$ that satisfies Eq. (5), implying Theorem 2 cannot guarantee that $\tau = \{\tau_1, \tau_2, \tau_3\}$ is schedulable by FP on 10 processors.

Motivated by Example 1, we would like to address the following issues for *non-parallel execution constrains* for the proposed RTA framework: (Step 1) how to express a constraint regarding concurrent execution for two tasks, and how to incorporate the constraint into the RTA framework, (Step 2) how to generalize the expression and incorporation of constraints for more than two tasks, and (Step 3) how to develop a systematical way to fully utilize the constraints to improve RTA, including finding a set of tasks subject to each constraint.

We first focus on two tasks that cannot be executed at the same time in the following lemma.

Lemma 3: If $m_i + m_j > m$ holds for τ_i and τ_j ($\ne \tau_i$), then the following inequality holds.

$$I_{k\leftarrow i}(L) + I_{k\leftarrow j}(L) \le L - C_k + 1 \tag{9}$$

Proof: Suppose that Eq. (9) is violated even though $m_i + m_j > m$ holds. By the definition of $I_{k\leftarrow i}(L)$ and $I_{k\leftarrow j}(L)$, each of them is no larger than $(L - C_k + 1)$. By the pigeonhole principle, the supposition implies there exists at least one k-interference time slot where a job of τ_i and a job of τ_k are executed at the same time. This contradicts $m_i + m_j > m$ in the supposition, which means no concurrent execution of any job of τ_i and any job of τ_j. ■

In Example 1, while Eq. (7) for τ_3 calculates interference upper-bounds $I^+_{3\leftarrow 1}(L)$ and $I^+_{3\leftarrow 2}(L)$ for $C_3{=}2 \le L \le D_3{=}5$, as $W_1(L) = L$ and $W_2(L) = L$, respectively, Lemma 3 proves that those upper-bounds are over-estimated because $W_1(L) + W_2(L) = 2 \cdot L$ is strictly larger than $(L - C_3 + 1) = L$. Using the lemma, we can reduce the upper-bound of the numerator of the fraction in Eq. (5) (i.e,. sum of $A_{k\leftarrow j}(L)$), as follows.

Lemma 4: If $m_i + m_j > m$ holds for τ_i and τ_j ($\ne \tau_i$) and $m_i \ge m_j$, then the following inequality holds.

$$
\begin{aligned}
& A_{k\leftarrow i}(L) + A_{k\leftarrow j}(L) \\
\leq \quad & I_{k\leftarrow i}(L)\cdot \min\big(m_i, m-m_k+1\big) \\
& + I_{k\leftarrow j}(L)\cdot \min\big(m_j, m-m_k+1\big) \\
\leq \quad & I^{+}_{k\leftarrow i}(L)\cdot \min\big(m_i, m-m_k+1\big) \\
& + \widehat{I_{k\leftarrow j}(L)}\cdot \min\big(m_j, m-m_k+1\big), \qquad (10)
\end{aligned}
$$

where $\widehat{I_{k\leftarrow j}(L)}$ is set to the minimum between (a) $I^{+}_{k\leftarrow j}(L)$ and (b) $\big((L-C_k+1)-I^{+}_{k\leftarrow i}(L)\big)$. Recall that $A_{k\leftarrow i}(L)$ and $I_{k\leftarrow i}(L)$ are the *actual* amount of execution (interference) and the *actual* duration of interference, respectively, while $I^{+}_{k\rightarrow i}(L)$ is an *upper bound* of the duration of interference.

Proof: Since the first inequality is proved in Lemma 2, we now prove the second inequality with two cases: Case 1 where $I^{+}_{k\leftarrow i}(L)+I^{+}_{k\leftarrow j}(L) \leq L-C_k+1$ holds, and its opposite Case 2.

(Case 1) In this case, $\widehat{I_{k\leftarrow j}(L)}$ is set to (a); therefore, the LHS of Eq. (10) is simply upper-bounded by the RHS, by considering the definition of the actual interference terms and their upper-bounds.

(Case 2) Let $I'_{k\leftarrow i}(L)$ ($\leq$ $I^{+}_{k\leftarrow i}(L)$) and $I'_{k\leftarrow j}(L)$ ($\leq$ $I^{+}_{k\leftarrow j}(L)$) denote new interference upper-bounds for $I_{k\leftarrow i}(L)$ and $I_{k\leftarrow j}(L)$, respectively. As they are interference upper-bounds, we can apply Eq. (9), meaning that the new constraint $I'_{k\leftarrow i}(L)+I'_{k\leftarrow j}(L) \leq L-C_k+1$ holds. We need to calculate a safe upper-bound of the LHS of Eq. (10), using the new constraint. Since $m_i \geq m_j$, the upper-bound is maximized when $I'_{k\leftarrow i}(L)$ is the largest, which is equal to $I^{+}_{k\leftarrow i}(L)$. In this case, $I'_{k\leftarrow j}(L)$ is the smallest, which is (b) by applying the new constraint. Therefore, the lemma holds. ∎

Applying Lemma 4 to τ_3 in Example 1, we can upper-bound the LHS of Eq. (10) for $\{\tau_1,\tau_2\}$ as $W_1(L)\cdot m_1 + \big((L-C_3+1)-W_1(L)\big)\cdot m_2 = L\cdot 6 + (L-L)\cdot 5 = L\cdot 6$, instead of $W_1(L){\cdot}m_1 + W_2(L){\cdot}m_2 = L{\cdot}6 + L{\cdot}5 = L{\cdot}11$. Then, Eq. (5) for τ_3 is calculated as $C_3+\big\lfloor \frac{L\cdot 6}{m-m_3+1}\big\rfloor = 1+\big\lfloor\frac{L\cdot 6}{9}\big\rfloor \leq L$ for every $C_3{=}2 \leq L \leq D_3{=}5$, meaning that Theorem 2 in conjunction with Lemma 4 guarantees $\tau = \{\tau_1,\tau_2,\tau_3\}$ is schedulable by FP on a 10-processor platform.

We now generalize Lemmas 3 and 4 (focusing on only two tasks) to the situation where there are g (≥ 2) tasks in τ' in which any h ($\leq g$) of them cannot be executed at the same time. The following example presents the situation where $g = 3$ and $h = 3$.

Example 2: Recall the task set in Example 1. Dividing τ_1 into two tasks, we consider a set of four gang tasks executed on $m = 10$ processors: $\tau_{1a}(T_{1a}{=}10, D_{1a}{=}10, C_{1a}{=}5, m_{1a}{=}3)$, $\tau_{1b}(10,10,5,3)$, $\tau_2(10,10,5,5)$, and $\tau_3(5,5,1,2)$. We apply FP scheduling where the priority of τ_{1a} and that of τ_3 are the highest and lowest, respectively. Then, among three jobs of tasks in $\tau' = \{\tau_{1a},\tau_{1b},\tau_2\}$, any two of them can be executed in parallel (because of $m_{1a}{+}m_2{=}3{+}5{=}8 \leq m{=}10$), while the three jobs cannot be executed at the same time (because of $m_{1a}{+}m_{1b}{+}m_2{=}3{+}3{+}5{=}11 > m{=}10$). Since any two of jobs of tasks in τ' can be executed at the same time, tasks in τ' cannot miss their job deadlines for any job arrival pattern invoked by the tasks (such as two patterns in Figure 1 with splitting τ_1 into τ_{1a} and τ_{1b}). Also, since the three jobs of tasks in τ' cannot be concurrently executed, at least two (i.e,. $10{-}5{-}3{=}2$) processors are not occupied by $\{\tau_{1a},\tau_{1b},\tau_2\}$ in any time, meaning that τ_3 does not miss its job deadlines.

However, we cannot apply Lemmas 3 and 4 to yield tight response time calculation of τ_3, because there is no pair of two tasks τ_i and τ_j with $m_i{+}m_j > m$ among $\{\tau_{1a},\tau_{1b},\tau_2\}$. Therefore, Theorem 2 for τ_3 calculates $I^{+}_{3\leftarrow 1a}(L)$, $I^{+}_{3\leftarrow 1b}(L)$ and $I^{+}_{3\leftarrow 2}(L)$ independently as $W_{1a}(L) = W_{1b}(L) = W_2(L) = L$ for every $C_3{=}2 \leq L \leq D_3{=}5$, yielding Eq. (5) for τ_3 as $C_3 + \big\lfloor \frac{L\cdot 3+L\cdot 3+L\cdot 5}{m-m_3+1}\big\rfloor = 1 + \big\lfloor\frac{L\cdot 11}{9}\big\rfloor \geq 1 + L$, which is the same result as Example 1. Therefore, Theorem 2 cannot guarantee that $\tau = \{\tau_{1a},\tau_{1b},\tau_2,\tau_3\}$ is schedulable by FP on 10 processors.

Motivated by the example, we extend Lemma 3 for g (≥ 2) and h ($\leq g$), such that there are a set of g tasks in τ', in which any h tasks of the set cannot be executed at the same time.

Lemma 5: If there exists a set of g (≥ 2) tasks $\tau' \subset \tau$ ($\tau_k \notin \tau'$) such that $\sum_{h \text{ tasks } \tau_i\in\tau'} m_i > m$ holds for any h ($\leq g$) tasks in τ', then the following inequality holds.

$$
\sum_{\tau_i\in\tau'} I_{k\leftarrow i}(L) \leq (h-1)\cdot(L-C_k+1) \qquad (11)
$$

Proof: The proof is similar to that of Lemma 3. Suppose that Eq. (11) is violated even though the "if" statement of the lemma is true. By Definition 1, $I_{k\leftarrow i}(L)$ for $\tau_i \in \tau'$ is not larger than $(L-C_k+1)$. By the pigeonhole principle, the supposition implies there exists at least one k-interference time slot where at least h jobs of tasks in τ' are executed. This contradicts $\sum_{h \text{ tasks } \tau_i\in\tau'} m_i > m$ holds for any h tasks in τ' in the supposition, which means that there is no concurrent execution of at least h jobs of tasks in τ'. ∎

Applying Lemma 5, we can generalize Lemma 4 as follows.

Theorem 3: If there exists a set of g (≥ 2) tasks $\tau' \subset \tau$ ($\tau_k \notin \tau'$) such that $\sum_{h \text{ tasks } \tau_i\in\tau'} m_i > m$ holds for any h ($\leq g$) tasks in τ', then the following inequality holds.

$$
\begin{aligned}
\sum_{\tau_i\in\tau'} A_{k\leftarrow i}(L) \leq & \sum_{\tau_i\in\tau'} I_{k\leftarrow i}(L)\cdot \min\big(m_i, m-m_k+1\big) \\
\leq & \sum_{\tau_i\in\tau'} \widehat{I_{k\leftarrow i}(L)}\cdot \min\big(m_i, m-m_k+1\big), \quad (12)
\end{aligned}
$$

where $\widehat{I_{k\leftarrow i}(L)}$ is assigned as follows, assuming tasks τ_i in τ' are indexed from τ_1 to τ_g and sorted in a descending order of m_i, without loss of generality.

$$
\begin{cases}
I^{+}_{k\leftarrow i}(L), & \text{if } \mathcal{C}1 \text{ holds;} \\
(h-1)\cdot(L-C_k+1) - \sum_{z=1}^{i-1} I^{+}_{k\leftarrow z}(L), & \text{if } \mathcal{C}2 \text{ holds;} \\
0, & \text{otherwise.}
\end{cases}
$$

Note that $\mathcal{C}1$ is $\sum_{z=1}^{i} I^{+}_{k\leftarrow z}(L) \le (h-1)\cdot(L-C_k+1)$. Also, $\mathcal{C}2$ is $\sum_{z=1}^{i-1} I^{+}_{k\leftarrow z}(L) \le (h-1)\cdot(L-C_k+1)$ and $\sum_{z=1}^{i} I^{+}_{k\leftarrow z}(L) > (h-1)\cdot(L-C_k+1)$.

Proof: Since the first inequality is proved in Lemma 2, we now prove the second inequality with two cases: Case 1 where $\sum_{\tau_i\in\tau'} I^{+}_{k\leftarrow i}(L) \le (h-1)\cdot(L-C_k+1)$ holds, and its opposite Case 2.

(Case 1) In this case, $\widehat{I_{k\leftarrow i}(L)}$ for every task $\tau_i \in \tau'$ is set to $I^{+}_{k\leftarrow i}(L)$ because $\mathcal{C}1$ holds for every task $\tau_i \in \tau'$. Therefore, the LHS of Eq. (12) is simply upper-bounded by the RHS, by considering the definition of the actual interference duration $I_{k\leftarrow i}(L)$ and their upper-bounds $I^{+}_{k\leftarrow i}(L)$.

(Case 2) The remaining proof is similar to that of Case 2 of Lemma 4, and aims at maximizing $\sum_{\tau_i\in\tau'} I'_{k\leftarrow i}(L)\cdot \min\big(m_i, m-m_k+1\big)$ for new interference upper-bounds $\{I'_{k\leftarrow i}(L)\}_{\tau_i\in\tau'}$ such that $I'_{k\leftarrow i}(L) \le I^{+}_{k\leftarrow i}(L)$ is satisfied for every $\tau_i \in \tau'$ and Eq. (11) for $\{I'_{k\leftarrow i}(L)\}_{\tau_i\in\tau'}$ holds. To this end, we assign the maximum budget for $I'_{k\leftarrow i}(L)$ (which is $I^{+}_{k\leftarrow i}(L)$) to a set of multiple τ_is that have the largest m_i (satisfying $\mathcal{C}1$), the remaining budget to a single τ_i that has the next largest m_i (satisfying $\mathcal{C}2$), and no budget to remaining τ_is. Therefore, the LHS of Eq. (12) with any $\{I'_{k\leftarrow i}(L)\}_{\tau_i\in\tau'}$ is upper-bounded by the RHS. ■

Applying Theorem 3 to τ_3 in Example 2, we can upper-bound of the LHS of Eq. (12) for $\tau' = \{\tau_{1a}, \tau_{1b}, \tau_2\}$ as $W_2(L)\cdot m_2 + W_{1a}(L)\cdot m_{1a} + \big(2\cdot(L-C_3+1) - W_2(L) - W_{1a}(L)\big)\cdot m_{1b} = L\cdot 5 + L\cdot 3 + 0\cdot 3 = L\cdot 8$, instead of $W_{1a}(L)\cdot m_{1a} + W_{1b}(L)\cdot m_{1b} + W_2(L)\cdot m_2 = L\cdot 3 + L\cdot 3 + L\cdot 5 = L\cdot 11$. Then, Eq. (5) for τ_3 is calculated by $C_3 + \left\lfloor \frac{L\cdot 8}{m-m_3+1} \right\rfloor = 1 + \left\lfloor \frac{L\cdot 8}{9} \right\rfloor \le L$ for every $C_3=2 \le L \le D_3=5$, meaning that Theorem 1 in conjunction with Theorem 3 guarantees $\tau = \{\tau_{1a}, \tau_{1b}, \tau_2, \tau_3\}$ is schedulable by FP on 10 processors.

The remaining issue is how to find τ' that can efficiently utilize Theorem 3. Algorithm 2 explains the overall process of calculating $A_k^{\text{total}}(R_k)$ (i.e., an upper-bound of the numerator of the fraction in Eq. (5) for given τ_k) by utilizing Theorem 3, which can replace Line 6 of the RTA framework in Algorithm 1.

Algorithm 2 finds τ' such that $\sum_{h \text{ tasks } \tau_i\in\tau'} m_i > m$ holds for any h tasks, and then applies Theorem 3 to τ'. This process is performed from $h=2$, and the first index for each τ' that shares the same h is set to $x^*=1$ (Line 2). For each task τ_x sorted in a descending order of m_x, Lines 4–6 find the case where we will not apply Lemma 5. If the reason is due to the insufficient number of target tasks in $\tau' = \{\tau_i\}_{i=x^*}^{x}$ for current h, we include τ_x in τ' by "continue" (Line 4). If the reason is due to the insufficient parallelism of tasks in τ', we increase h and include τ_x in τ' by "continue" (Line 5). If the reason is τ_x is not the last task in τ' (checked by the sum of m_y of h tasks with the smallest m_y including the current task (τ_x) and next task (τ_{x+1})), we include τ_x in τ' by "continue" (Line 6). If all the "if" statements in Lines 4–6 are not satisfied, Line 7 applies Lemma 5. If Eq. (11) in the lemma is violated, we apply Theorem 3, set the first index of the next τ' to $x^* = x+1$, and increase h; otherwise, we keep the current τ' without applying Theorem 3 and increase h.

Algorithm 2 Calculation of $A_k^{\text{total}}(R_k)$ in Algorithm 1 for given $\{I^{+}_{k\leftarrow i}(L)\}$, by addressing non-parallel execution constraints

```
1: Assuming {τx ∈ τ | τx ≠ τk} are sorted in a descending order of
   m_x, index them from 1 to n − 1 (where n is τk's index)
2: h ← 2, x* ← 1, and A_k^total(R_k) ← 0
3: for x = 1, 2, 3, ..., n − 1 do
4:     if x − x* + 1 < h, then continue
5:     if Σ_{y=x*}^{x} m_y ≤ m, then h ← h + 1 and continue
6:     if x ≠ n − 1 and Σ_{y=x−h+2}^{x+1} m_y > m, then continue
7:     if Eq. (11) with {I_{k←i}(L) = I^+_{k←i}(L)} does not hold for
       τ' = {τy | x* ≤ y ≤ x} with L = R_k, then A_k^total(R_k) ←
       A_k^total(R_k)+ the RHS of Eq. (12) for τ' with L = R_k, x* ←
       x + 1, h ← h + 1
8:     else h ← h + 1
9: end for
10: for x = x*, x* + 1, ..., n − 1 do
11:     A_k^total(R_k) ← A_k^total(R_k) + I^+_{k←x}(R_k) · min(m_x, m − m_k + 1)
12: end for
```

If we apply Algorithm 2 to the task set in Example 2 on 10 processors, τ_2, τ_{1a} and τ_{1b} are sequentially investigated as τ_x, when $\tau_k = \tau_3$ (by Line 1). For $\tau_x = \tau_2$, Line 4 executes "continue." For $\tau_x = \tau_{1a}$, $m_2 + m_{1a} = 5+3 \le m = 10$ holds, so Line 5 executes $h \leftarrow 3$ and "continue". Finally, for $\tau_x = \tau_{1b}$, Line 7 confirms the violation of Eq. (11), and applies Theorem 3. As a result, $C_3 + \left\lfloor \frac{L\cdot5+L\cdot3+L\cdot0}{m-m_3+1} \right\rfloor = 1 + \left\lfloor \frac{L\cdot 8}{9} \right\rfloor \le L$ holds for $C_3=2 \le L \le D_3=5$, yielding schedulability of τ_3.

Using Algorithm 2, we can develop a tighter schedulability test for FP and EDF.

Theorem 4: Algorithm 1 by replacing Line 6 with Algorithm 2 yields the schedulability analysis for FP and EDF, if we apply the RHS of Eq. (7) and that of Eq. (8) for $I^{+}_{k\leftarrow i}(L)$, respectively.

Proof: By Algorithm 1, Lemma 2, and Eqs. (7) and (8), the remaining proof is to prove that $A_k^{\text{total}}(R_k)$ calculated by Algorithm 2 is an upper-bound of $\sum_{\tau_i\in\tau} A_{k\leftarrow i}(R_k)$.

Regarding an upper-bound of $\sum_{\tau_i\in\tau} A_{k\leftarrow i}(R_k)$, the RHS of Eq. (12) is applied to individual τ's in Line 7, and $I^{+}_{k\leftarrow x}(R_k)\cdot \min(m_x, m-m_k+1)$ is applied to tasks that do not belong to individual τ's (i.e., a set of tasks in Line 10); the former and the latter are proven to be a safe upper-bound by Theorem 3 and Lemma 2, respectively. Therefore, it suffices to prove that (i) each task belongs to either a set of tasks in Line 10 or only one τ' and (ii) every τ' satisfies the supposition of Theorem 3.

In Lines 4–8, x is not changed. Line 7 is the only line that completes the current τ' and generates a next τ' by assigning $x^* = x+1$. Therefore, the first index that does not belong to any τ' is x^* after Lines 2–9. Therefore, (i) holds.

Line 4 guarantees $h \le g$, where g is the number of tasks in τ'. Since tasks are sorted in descending order of m_x in Line 1, Line 6 checks whether the h tasks with the smallest m_x satisfy $\sum_{\text{the } h \text{ tasks}} m_x > m$ if we add the current task (τ_x) and the next task (τ_{x+1}) to the current τ'. Therefore, Line 6 along with Line 5 guarantees $\sum_{h \text{ tasks } \tau_i \in \tau'} m_i > m$ for any τ' to be performed in Line 7. Therefore, (ii) holds. ■

V. Addressing Over-Estimation of k-interference Processor Occupation

In this section, we focus on the issue of *over-estimation of k-interference processor occupation* by the proposed RTA framework. Suppose that a job of τ_i (J_i) and a job of τ_j (J_j) are executed in a k-interference time slot. If $(m_i{+}m_j > m{-}m_k{+}1)$ holds, the amount of execution of J_i and J_j in the k-interference slot (which is larger than $(m{-}m_k{+}1)$) cannot fully contribute to the amount of execution on k-interference processors in the k-interference slot (which is exactly $(m{-}m_k{+}1)$ by Definition 3). In this situation, Algorithm 1 with reuse of $I^+_{k\leftarrow i}(L)$ yields over-estimation of k-interference occupation, as follows.

Example 3: We consider a set of four gang tasks executed on $m = 10$ processors: $\tau_1(T_1{=}10, D_1{=}10, C_1{=}9, m_1{=}4)$, $\tau_2(10, 10, 9, 3)$, $\tau_3(10, 10, 9, 2)$, and $\tau_4(10, 10, 1, 3)$. We apply FP scheduling where the priority of τ_1 and that of τ_4 are the highest and lowest, respectively. We can confirm that τ_1, τ_2 and τ_3 are schedulable for any job arrival pattern, because the sum of their m_i (4+3+2) is not larger than the number of processors (10), as shown in Figure 2.

Apart from actual results, if we apply Theorem 2 for τ_4, the interference term for τ_1 ($I^+_{4\leftarrow 1}(L)$), τ_2 ($I^+_{4\leftarrow 2}(L)$) and τ_3 ($I^+_{4\leftarrow 3}(L)$) are added to the numerator of the fraction in Eq. (5), yielding no guarantee of the schedulability of τ_4. For example, in case of $L = D_4$, since $W_1(D_4) = W_2(D_4) = W_3(D_4) = 9$, the numerator of the fraction in Eq. (5) is $9{\cdot}4{+}9{\cdot}3{+}9{\cdot}2 = 81$, implying $C_4 + \lfloor \frac{81}{m-m_4+1} \rfloor = 1 + \lfloor \frac{81}{8} \rfloor = 11 > 10$; other $C_4 \le L \le D_4$ also cannot guarantee the schedulability of τ_4. However, considering $m_1 + m_2 + m_3 = 9 > m - m_4 + 1 = 8$ holds, if jobs of τ_1, τ_2 and τ_3 are concurrently executed at a time slot, some of the amount of their execution (i.e., some of 9) should not be executed on the 4-interference processors (whose number is $10{-}3{+}1{=}8$); recall Definition 3 that indicates the existence of exactly $(m{-}m_k{+}1)$ k-interference processors in each k-interference time slot. Next, in any case where each job of τ_1, τ_2 and τ_3 executes for 9 time units in $[0, 10)$, all the three jobs should be executed at the same time in at least 7 time units, e.g., Fig. 2(b). In each of those time slots, one amount of execution (marked by "X") cannot be executed on the 4-interference processors, which should be deducted to the amount of interference of τ_1, τ_2 and τ_3 on τ_4. Considering those interference deductions, τ_4, in reality, is schedulable in any case (e.g., adding a job of τ_4's execution to the two cases in Fig. 2 is possible), which is different from the schedulability analysis result.

Motivated by Example 3, the following lemma develops how to deduct over-estimation of k-interference processor occupation, based on the calculation of I^Δ_k, the minimum number of k-interference time slots in which $I^+_{k\leftarrow i}(L)$ for *every* $\tau_i \in \tau'$ contributes to $\sum_{\tau_i\in\tau'} A_{k\leftarrow i}(L)$.

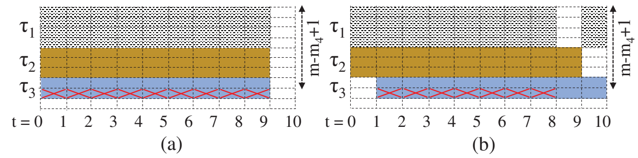


Fig. 2. Schedules of τ_1, τ_2, and τ_3 in Example 3 by FP on 10 processors: (a) when every job of all the three tasks are released at $t = 0$, and (b) a job of τ_1, that of τ_2 and that of τ_3 are periodically released from $t = -1$, $t = 0$, and $t = 1$, respectively. The execution marked as "X" cannot contribute to the amount of interference on τ_4 with $m_4 = 3$.

Lemma 6: The following inequality holds for $\tau' \subset \tau$ ($\tau_k \notin \tau'$), if τ' satisfies both $\sum_{\tau_i\in\tau'} \min(m_i, m - m_k + 1) > m - m_k + 1$ and $I^\Delta_k > 0$.

$$\sum_{\tau_i\in\tau'} A_{k\leftarrow i}(L) \le \sum_{\tau_i\in\tau'} I^+_{k\leftarrow i}(L)\cdot \min(m_i, m - m_k + 1) - I^\Delta_k \cdot \Big(\sum_{\tau_i\in\tau'} \min(m_i, m - m_k + 1) - (m - m_k + 1)\Big), \quad (13)$$

where $I^\Delta_k = (L - C_k + 1) - \sum_{\tau_i\in\tau'} \big((L - C_k + 1) - I^+_{k\leftarrow i}(L)\big)$.

Proof: We focus on $(L{-}C_k{+}1)$ k-interference time slots of the target interval of length L. We first prove that there exists at least I^Δ_k k-interference time slots in which $I^+_{k\leftarrow i}(L)$ for *every* $\tau_i \in \tau'$ contributes to $\sum_{\tau_i\in\tau'} A_{k\leftarrow i}(L)$. Since $I^+_{k\leftarrow i}(L) \le (L - C_k + 1)$ holds by definition, $\big((L - C_k + 1) - I^+_{k\leftarrow i}(L)\big)$ is the number of k-interference time slots where τ_i does not contribute to $\sum_{\tau_i\in\tau'} A_{k\leftarrow i}(L)$. By disallowing any overlap of the set of $\big((L - C_k + 1) - I^+_{k\leftarrow i}(L)\big)$ time slots for every $\tau_i \in \tau'$, $\sum_{\tau_i\in\tau'} \big((L - C_k + 1) - I^+_{k\leftarrow i}(L)\big)$ (denoted by X) implies an upper-bound of the number of k-interference time slots in which at least one $\tau_i \in \tau'$ do not contribute to $\sum_{\tau_i\in\tau'} A_{k\leftarrow i}(L)$. This means, $(L{-}C_k{+}1{-}X)$, which is equal to I^Δ_k, is a lower-bound of the number of k-interference time slots in which $I^+_{k\leftarrow i}(L)$ for *every* $\tau_i \in \tau'$ contributes to $\sum_{\tau_i\in\tau'} A_{k\leftarrow i}(L)$.

Considering $\sum_{\tau_i\in\tau'} \min(m_i, m - m_k + 1) > m - m_k + 1$ in the supposition of the lemma, the first part implies that there should exist at least I^Δ_k k-interference time slots in which the amount of execution of jobs of tasks in τ' is larger than $(m{-}m_k{+}1)$. Considering the definition of the k-interference processors in Definition 3, there should exist at least I^Δ_k k-interference time slots, in each of which at least $\sum_{\tau_i\in\tau'} \min(m_i, m - m_k + 1) - (m - m_k + 1)$ amount of execution of jobs of tasks in τ' is not executed on k-interference processors. This means, $I^\Delta_k \cdot \big(\sum_{\tau_i\in\tau'} \min(m_i, m - m_k + 1) - (m{-}m_k{+}1)\big)$ amount of execution of jobs of tasks in τ' cannot contribute to $\sum_{\tau_i\in\tau'} A_{k\leftarrow i}(L)$ by Definition 4. Therefore, we can deduct those amount as shown in Eq. (13). ■

If we apply Lemma 6 to $\tau_k = \tau_4$, $\tau' = \{\tau_1, \tau_2, \tau_3\}$ and $L = D_4 = 10$ of Example 3, $I^\Delta_k = 10 - (10-9) - (10-9) - (10-9) = 7$ holds. Therefore, $7\cdot\big(4{+}3{+}2{-}(10{-}3{+}1)\big) = 7$ amount of execution should be deducted from $9\cdot4 + 9\cdot3 + 9\cdot2 = 81$, and then $C_4 + \left\lfloor \frac{81-7}{m-m_4+1} \right\rfloor = 1 + \left\lfloor \frac{74}{8} \right\rfloor = 10$ holds, implying τ_4 is schedulable.

We now generalize Lemma 6 in order to more tightly deduct over-estimation of k-interference processor occupation.

Theorem 5: The following inequality holds for $\tau' \subset \tau$ ($\tau_k \notin \tau'$).

$$\sum_{\tau_i \in \tau'} A_{k \leftarrow i}(L) \leq \sum_{\tau_i \in \tau'} I^{+}_{k \leftarrow i}(L) \cdot \min(m_i, m - m_k + 1) - A^{\Delta}_k \tag{14}$$

where $A^{\Delta}_k = \sum_{x=1}^{|\tau'|} A^{\Delta}_{k \leftarrow x}$, assuming $\tau_x \in \tau'$ are indexed from τ_1 to $\tau_{|\tau'|}$, sorted in any given order. Let $I^{\Delta}_k(i)$ denote $(L - C_k + 1) - \sum_{x=1}^{i} \big((L - C_k + 1) - I^{+}_{k \leftarrow x}(L)\big)$, and $m^{\text{sum}}(i)$ denote $\sum_{x=1}^{i} \min(m_x, m - m_k + 1)$. Then, $A^{\Delta}_{k \leftarrow x}$ is calculated sequentially from τ_1 to $\tau_{|\tau'|}$, by $A^{\Delta}_{k \leftarrow x} =$

$$\begin{cases} I^{\Delta}_k(i) \cdot \min(m_i, m - m_k + 1), & \\ \qquad \text{if } m^{\text{sum}}(i-1) > m - m_k + 1 \text{ and } I^{\Delta}_k(i) > 0; \\ I^{\Delta}_k(i) \cdot \big(m^{\text{sum}}(i) - (m - m_k + 1)\big), & \\ \qquad \text{else if } m^{\text{sum}}(i) > m - m_k + 1 \text{ and } I^{\Delta}_k(i) > 0; \\ 0, \qquad \text{otherwise.} \end{cases} \tag{15}$$

Proof: We consider the amount of interference deduction for (Case 1) "if", (Case 2) "else if" and (Case 3) "otherwise" conditions for Eq. (15).

Case 2 holds by applying $\tau' = \{\tau_x\}_{x=1}^{i}$ to Lemma 6, and Case 3 does not deduct the amount of interference. Therefore, we need to prove Case 1.

The "if" statement in Case 1 implies that we already have a set of $i-1$ tasks $\{\tau_x\}_{x=1}^{i-1}$ that satisfies $\sum_{x=1}^{i-1} \min(m_x, m - m_k + 1) > m - m_k + 1$ without including τ_i itself. If we apply the first part of the proof of Lemma 6, there exists at least $I^{\Delta}_k(i)$ k-interference time slots where all tasks $\{\tau_x\}_{x=1}^{i}$ contribute to $\sum_{\tau_i \in \tau'} A_{k \leftarrow i}(L)$. Considering $\sum_{x=1}^{i-1} \min(m_x, m - m_k + 1) > m - m_k + 1$ holds without adding τ_i, $I^{\Delta}_k \cdot m_i$ is an additional amount of execution of jobs of tasks in τ' that cannot contribute to $\sum_{\tau_i \in \tau'} A_{k \leftarrow i}(L)$, which proves Case 1. ■

Algorithm 3 explains the overall process for calculating $A^{\text{total}}_k(R_k)$ (i.e., an upper-bound of the numerator of the fraction in Eq. (5) for given τ_k) by utilizing Theorem 5, which can replace Line 6 of the RTA framework in Algorithm 1. In Lines 1–4, we calculate $A^{\text{total}}_k(R_k)$ using the known upper-bounds of the duration of interference $\{I^{+}_{k \leftarrow i}(R_k)\}$. We sort tasks $\{\tau_i \in \tau | \tau_i \neq \tau_k\}$ in a given order[2] (Line 5). Then, for each task, we check whether it can be included in τ' for applying Theorem 5 (Lines 7–12). In Lines 8–11, I^{Δ}_k and m^{sum} respectively correspond to $I^{\Delta}_k(i)$ and $m^{\text{sum}}(i)$ in Eq. (15). If the "if" statement in Line 8 holds (meaning that $I^{\Delta}_k(i) \leq 0$ in Eq. (15)), we do not include τ_i in τ' by not executing

[2]While we can apply any sorting order (but it yields more/less tight calculation of A^{Δ}_k), we apply a descending order of $\big((R_k - C_k + 1) - I^{+}_{k \leftarrow i}(R_k)\big)/m_i$, a reasonable heuristic.

Algorithm 3 Calculation of $A^{\text{total}}_k(R_k)$ in Algorithm 1 for given $\{I^{+}_{k \leftarrow i}(L)\}$, by addressing the over-estimation issue

1: $A^{\text{total}}_k(R_k) \leftarrow 0$
2: **for** $\tau_i \in \tau | \tau_i \neq \tau_k$ **do**
3: $\quad A^{\text{total}}_k(R_k) \leftarrow A^{\text{total}}_k(R_k) + I^{+}_{k \leftarrow i}(R_k) \cdot \min(m_i, m - m_k + 1)$
4: **end for**
5: Assuming $\{\tau_i \in \tau | \tau_i \neq \tau_k\}$ are sorted in any given order.
6: $A^{\Delta}_k \leftarrow 0$, $I^{\Delta}_k \leftarrow R_k - C_k + 1$, $m^{\text{sum}} \leftarrow 0$
7: **for** $\tau_i \in \tau | \tau_i \neq \tau_k$ **do**
8: $\quad$ **if** $I^{\Delta}_k - \big((R_k - C_k + 1) - I^{+}_{k \leftarrow i}(R_k)\big) \leq 0$ **then** continue
9: $\quad I^{\Delta}_k \leftarrow I^{\Delta}_k - \big((R_k - C_k + 1) - I^{+}_{k \leftarrow i}(R_k)\big)$, $m^{\text{sum}} \leftarrow m^{\text{sum}} + \min(m_i, m - m_k + 1)$
10: $\quad$ **if** $m^{\text{sum}} - \min(m_i, m - m_k + 1) > m - m_k + 1$ **then** $A^{\Delta}_k \leftarrow A^{\Delta}_k + I^{\Delta}_k \cdot \min(m_i, m - m_k + 1)$
11: $\quad$ **else if** $m^{\text{sum}} > m - m_k + 1$ **then** $A^{\Delta}_k \leftarrow A^{\Delta}_k + I^{\Delta}_k \cdot \big(m^{\text{sum}} - (m - m_k + 1)\big)$
12: **end for**
13: $A^{\text{total}}_k(R_k) \leftarrow A^{\text{total}}_k(R_k) - A^{\Delta}_k$

Line 9. Otherwise, we include τ_i in τ'. Then, Lines 10 and 11 correspond to the "if" and "else if" cases in Eq. (15), and we add the corresponding $A^{\Delta}_{k \leftarrow i}$ in Eq. (15) to A^{Δ}_k. Finally, Line 13 deducts A^{Δ}_k from $A^{\text{total}}_k(R_k)$.

Using Algorithm 3, we can develop a tighter schedulability test for FP and EDF.

Theorem 6: Algorithm 1 by replacing Line 6 with Algorithm 3 yields the schedulability analysis for FP and EDF, if we apply the RHS of Eq. (7) and that of Eq. (8) for $I^{+}_{k \leftarrow i}(L)$, respectively.

Proof: By Algorithm 1, Lemma 2, and Eqs. (7) and (8), the remaining proof is to prove that $A^{\text{total}}_k(R_k)$ calculated by Algorithm 3 is an upper-bound of $\sum_{\tau_i \in \tau} A_{k \leftarrow i}(R_k)$.

Once we perform Algorithm 3, we can have τ' that consists of all $\{\tau_i \in \tau | \tau_i \neq \tau_k\}$ except tasks that satisfy the "if" condition in Line 8. After performing Line 9, we can easily check that I^{Δ}_k and m^{sum} respectively correspond to (i.e., equals to) $I^{\Delta}_k(i)$ and $m^{\text{sum}}(i)$ in Eq. (15). For τ', we can confirm that the "if" and "else if" cases of Eq. (15) respectively correspond to Lines 10 and 11. Also, the "otherwise" case of Eq. (15) corresponds to no line for the case where both "if" and "else if" conditions in Lines 10–11 are not satisfied. Therefore, by Theorem 5, Theorem 6 holds. ■

Utilizing Algorithm 2 or 3, we can address either the issue of non-parallel execution constraints or that of over-estimation of k-interference processor occupation, for the proposed RTA framework. Now, we combine the two techniques to address both issues at the same time, recorded in Algorithm 4. The idea is to apply Algorithm 2 and then Algorithm 3. In Line 1, we set new interference upper-bounds $\{I^{*}_{k \leftarrow i}(R_k)\}$ to be used for Algorithm 3, to $\{I^{+}_{k \leftarrow i}(R_k)\}$. In Line 2, we perform Algorithm 2 using the original upper-bounds $\{I^{+}_{k \leftarrow i}(R_k)\}$; meanwhile, for a subset τ' where the first technique in Theorem 3 is applied, we update $\{I^{*}_{k \leftarrow i}(R_k)\}$ as $\{\widehat{I_{k \leftarrow i}(R_k)}\}$ in Eq. (12). In Line 3, we apply the new interference upper-bounds $\{I^{*}_{k \leftarrow i}(R_k)\}$ to the second technique in order to deduct over-estimation by Lines 5–13 of Algorithm 3.

Algorithm 4 Calculation of $A_k^{\text{total}}(R_k)$ in Algorithm 1 for given $\{I^+_{k\leftarrow i}(L)\}$, by addressing the two issues together

1: Set $I^*_{k\leftarrow i}(R_k)$ to $I^+_{k\leftarrow i}(R_k)$ for every $\tau_i \in \tau$ ($\tau_i \notin \tau_k$).
2: Perform Algorithm 2 using $\{I^+_{k\leftarrow i}(R_k)\}$; meanwhile, if the "if" statement of Line 7 is true for τ', we set $I^*_{k\leftarrow i}(R_k)$ for every $\tau_i \in \tau'$, to $\widehat{I_{k\leftarrow i}(R_k)}$ in Eq. (12).
3: Perform Lines 5–13 of Algorithm 3 using $\{I^*_{k\leftarrow i}(R_k)\}$ instead of $\{I^+_{k\leftarrow i}(R_k)\}$.

Finally, applying Algorithm 4, we develop a tighter schedulability test for FP and EDF.

Theorem 7: Algorithm 1 by replacing Line 6 with Algorithm 4 yields the schedulability analysis for FP and EDF, if we apply the RHS of Eq. (7) and that of Eq. (8) for $I^+_{k\leftarrow i}(L)$, respectively.

Proof: By Algorithm 1, Lemma 2, and Eqs. (7) and (8), the remaining proof is to prove that $A_k^{\text{total}}(R_k)$ calculated by Algorithm 4 is an upper-bound of $\sum_{\tau_i\in\tau} A_{k\leftarrow i}(R_k)$.

Theorem 3, which is a basis of Algorithm 2, applies a "duration" constraint of Eq. (11) in Lemma 5 for $\{I^+_{k\leftarrow i}(L)\}$. To safely upper-bound $\sum_{\tau_i\in\tau} A_{k\leftarrow i}(L)$, Theorem 3 selects to include the largest m_i's first to each $I^+_{k\leftarrow i}(L)$ until the duration constraint holds, which is represented by $\widehat{I_{k\leftarrow i}(L)}$ in Eq. (12), where $\widehat{I_{k\leftarrow i}(L)} = 0$ means the corresponding $I^+_{k\leftarrow i}(L)$ is not selected. Therefore, it suffices to prove that any other selection cannot yield a larger upper-bound of $\sum_{\tau_i\in\tau} A_{k\leftarrow i}(L)$ in Algorithm 4 that performs Algorithm 2 in Line 2 and then Algorithm 3 in Line 3.

Let $\{\widehat{I'_{k\leftarrow i}(L)}\}$ denote a different selection from $\{\widehat{I_{k\leftarrow i}(L)}\}$. We will explain the case where $h = g = 2$ of Theorem 3, i.e., $m_h + m_j > m$; the proof of other cases is similar to this case. Suppose we exchange one time unit between τ_h and τ_j with $m_h > m_j$. That is, $\widehat{I'_{k\leftarrow h}(L)} = \widehat{I_{k\leftarrow h}(L)} - 1$ and $\widehat{I'_{k\leftarrow j}(L)} = \widehat{I_{k\leftarrow j}(L)} + 1$ holds; all other $\{\widehat{I'_{k\leftarrow i}(L)}\}$ are the same as $\{\widehat{I_{k\leftarrow i}(L)}\}$. Then, by the difference between m_h and m_j, $\min(m_h, m - m_k + 1) - \min(m_j, m - m_k + 1) \geq 0$ is a decrease of the total amount of interference by the new selection in Algorithm 2 (that utilizes Theorem 3), i.e., the RHS of Eq. (12) with $\{\widehat{I_{k\leftarrow i}(L)}\}$ subtracted by that with $\{\widehat{I'_{k\leftarrow i}(L)}\}$.

Therefore, the remaining step is to prove that $\min(m_h, m - m_k + 1) - \min(m_j, m - m_k + 1) \geq 0$ is an upper-bound of decrease of the total amount of interference *deduction* by the new selection in Algorithm 3 (that utilizes Theorem 5), i.e., the amount of interference deduction (A_k^{Δ}) with $\{\widehat{I_{k\leftarrow i}(L)}\}$ in the RHS of Eq. (14), subtracted by that with $\{\widehat{I'_{k\leftarrow i}(L)}\}$. Suppose that $\{\widehat{I_{k\leftarrow i}(L)}\}$ are assigned to processors in k-interference time slots. If we replace $\{\widehat{I_{k\leftarrow i}(L)}\}$ with $\{\widehat{I'_{k\leftarrow i}(L)}\}$, the only difference is the decrease of the amount of $\min(m_h, m - m_k + 1) - \min(m_j, m - m_k + 1)$ contribution on a *single* time slot, derived from $m_h + m_j > m$. Therefore, the optimal lower-bound (not its lower-bound from Theorem 5) A_k^{Δ} with $\{\widehat{I_{k\leftarrow i}(L)}\}$, subtracted by that with $\{\widehat{I'_{k\leftarrow i}(L)}\}$ is at most $\min(m_h, m - m_k + 1) - \min(m_j, m - m_k + 1)$. Therefore, $\min(m_h, m - m_k + 1) - \min(m_j, m - m_k + 1)$ is also an upper-bound of the difference between the lower-bound of A_k^{Δ} with $\{\widehat{I_{k\leftarrow i}(L)}\}$ from Theorem 5 and the optimal A_k^{Δ} with $\{\widehat{I'_{k\leftarrow i}(L)}\}$, which proves this step.

By repeating the exchange of one time unit between two tasks, we prove that any selection of $\{\widehat{I_{k\leftarrow i}(L)}\}$ subject to Eq. (11) cannot yield a larger upper-bound of $\sum_{\tau_i\in\tau} A_{k\leftarrow i}(L)$ in Algorithm 4 than the original selection. This proves the theorem. ■

Time-complexity. Since Algorithms 2 and 3 exhibit $O(n^2)$ and $O(n)$ time-complexity, respectively, Algorithm 4 exhibits $O(n^2)$ time-complexity, where n is the number of tasks in τ. This complexity exhibits one higher order than the corresponding part in the RTA framework that reuses existing $\{I^+_{k\leftarrow i}(L)\}$, i.e., $A_k^{\text{total}}(R_k)$ in Line 6 of Algorithm 1 is calculated by $\sum_{\tau_i\in\tau,\tau_i\neq\tau_k} I^+_{k\leftarrow i}(L) \cdot \min(m_i, m - m_k + 1)$ that requires $O(n)$. Therefore, Algorithm 1 by replacing Line 6 with Algorithm 4 yields $O\big(n^4 \cdot (\max D_i)^2\big)$, since the corresponding original RTA for the sequential task model is known to exhibit $O\big(n^3 \cdot (\max D_i)^2\big)$ time complexity [23].

VI. Evaluation

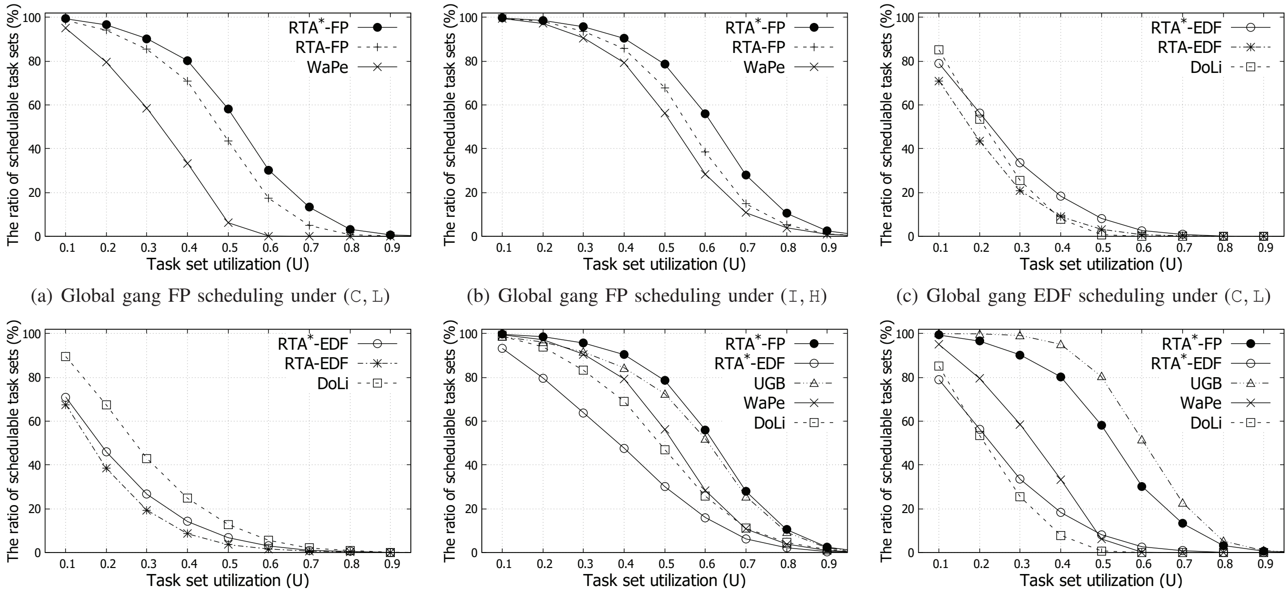
In this section, we compare our RTA framework designed for global gang scheduling, with existing schedulability tests for both global and non-global gang scheduling.

A. Evaluation Setting

The randomly generated task sets are based on [5]. For each number of processors m (i.e., 8, 16, 32, 64, 128 and 256), we consider four parameters: (S1) the type of the task set, i.e., implicit-deadline ($D_i = T_i$) and constrained-deadline ($D_i \leq T_i$), (S2) the distribution of task utilization $u_i \stackrel{\text{def.}}{=} C_i/T_i$, i.e., the binomial distribution with $p = 0.1, 0.3, 0.5, 0.7$ and 0.9,[3] (S3) the range of task parallelism m_i, i.e., $[1, \frac{1}{2}m]$ and $[1, m)$, and (S4) the range of task set utilization $U \stackrel{\text{def.}}{=} \sum_{\tau_i\in\tau} \frac{u_i \cdot m_i}{m}$, i.e., $[0.0, 0.1)$, $[0.1, 0.2)$, ... ,$[0.9, 1.0)$. For each task, the period T_i is uniformly selected in $[10ms, 1000ms]$; C_i is set to $u_i \cdot T_i$, where u_i is generated by S2; for implicit- and constrained-deadline tasks, D_i is set to T_i and uniformly selected in [C_i, T_i], respectively; and m_i is uniformly distributed in the range assigned by S3. For every combination of S1, S2, S3 and S4, we generate 1000 task sets, yielding $2\cdot5\cdot2\cdot10\cdot1000 = 200,000$ task sets in total for each m.

Using the generated sets, we compare our schedulability tests designed for preemptive *global* gang scheduling, with all existing schedulability tests for preemptive *global/non-global* gang scheduling subject to our task model (explained in Section II), as follows.

[3]For given p, task utilization is uniformly distributed in $[0.5, 1.0]$ and $[0.0, 0.5]$ with probability of p and $1.0-p$, respectively. Therefore, the average number of tasks in each task set decreases as p increases.



(a) Global gang FP scheduling under (C, L) (b) Global gang FP scheduling under (I, H) (c) Global gang EDF scheduling under (C, L)

(d) Global gang EDF scheduling under (C, H) (e) Any gang scheduling under (I, H) (f) Any gang scheduling under (C, L)

Fig. 3. Schedulability performance comparison of our schedulability tests with existing ones

- WaPe: global scheduling for FP in [10]
- DoLi: global scheduling for EDF in [5], [15]
- UGB: non-global scheduling (i.e., a generalization of partitioned scheduling) for FP in [11]
- RTA-FP and RTA-EDF: Theorem 2 for FP and EDF
- RTA^1-FP and RTA^1-EDF: Theorem 4 for FP and EDF
- RTA^2-FP and RTA^2-EDF: Theorem 6 for FP and EDF
- RTA*-FP and RTA*-EDF: Theorem 7 for FP and EDF

For fair comparison for the tests that employ FP as a prioritization policy, we apply DM (Deadline Monotonic) [24] to all the tests.

We count the number of task sets deemed schedulable by each of the above schedulability tests, and show the ratio of those task sets. We observe that the trend for the relative ratio among individual tests does not much vary with m. Therefore, we explain the representative results for $m = 64$ in the next subsections. For $m = 64$, we present the overall results for all generated task sets without any figure, *and* some interesting results with Fig. 3 for a subset of generated task sets subject to a pair of S1 and S3, denoted by $(\mathtt{I}/\mathtt{D}, \mathtt{L}/\mathtt{H})$, where I and D imply implicit-deadline and constrained-deadline task sets in S1, respectively, and L and H imply $m_i \in [1, \frac{1}{2}m]$ (i.e., low m_i) and $m_i \in [1, m)$ (i.e., high m_i), respectively. In each of Fig. 3, the X-axis represents the task set utilization U (i.e., S4), while the Y-axis represents the ratio of task sets deemed schedulable by each schedulability tests. Therefore, each point in Fig. 3 targets task sets subject to a given combination of S1, S3 and S4 while the target task sets include all parameters of S2.

B. Comparison of Global Gang Scheduling

As our RTA framework targets global gang scheduling, we now compare our RTA framework with a given prioritization policy, to an existing schedulability test for global gang scheduling with the same policy, i.e., RTA*-FP (and RTA-FP) versus WaPe, and RTA*-EDF (and RTA-EDF) versus DoLi.

For global gang FP scheduling, RTA*-FP and RTA-FP outperform WaPe under every combination of S1 and S3, and they respectively achieve 37.4% and 23.0% overall improvement over WaPe. This is because, while RTA-FP and WaPe share a similar schedulability analysis structure, RTA-FP tightly calculates a response time using the notion of k-interference slots/processors (and RTA*-FP more tightly does). The most favorable and unfavorable settings for RTA*-FP against WaPe, are $(\mathtt{C}, \mathtt{L})$ and $(\mathtt{I}, \mathtt{H})$, respectively, shown in Figs. 3(a) and (b); under the settings, RTA*-FP respectively finds 73.1% and 12.0% more schedulable task sets, compared to WaPe.

When it comes to global gang EDF scheduling, RTA*-EDF cannot outperform DoLi in that RTA*-EDF finds 13.1% less schedulable task sets than DoLi; however, the same does not hold under every setting. Since DoLi uses a notion of the maximum idle parallelism when a task cannot be executed, it is more effective for task sets with low m_i. Therefore, the performance of RTA*-EDF against DoLi varies with the setting for S3. For example, as shown in Figs. 3(c) and (d), RTA*-EDF finds 15.2% *more* and 31.3% *less* schedulable task sets compared to DoLi, respectively under $(\mathtt{C}, \mathtt{L})$ and $(\mathtt{C}, \mathtt{H})$.

In summary, RTA*-FP significantly outperforms the existing schedulability test that targets global gang FP scheduling, and RTA*-EDF complements the existing schedulability test for global gang EDF scheduling.

C. Comparison of Any Gang Scheduling

We now present the performance of our schedulability tests with all other existing ones, regardless of prioritization policies (i.e., EDF or FP) and scheduling categories (i.e., global or non-global). Overall, the schedulability ratio of $\texttt{RTA}^*\texttt{-EDF}$, `DoLi`, `WaPe`, and `UGB`, normalized by that of $\texttt{RTA}^*\texttt{-FP}$, is 52.2%, 60.1%, 72.8% and 105.0%, respectively. Between the two highest schedulability-performance tests $\texttt{RTA}^*\texttt{-FP}$ and `UGB`, we observe different schedulability performance behavior depending on the settings for S3. That is, under (`I`, `H`) and (`C`, `H`), $\texttt{RTA}^*\texttt{-FP}$ finds 5.2% and 0.3% *more* schedulable task sets than `UGB` (the former of which is shown in Fig. 3(e)). On the other hand, under (`I`, `L`) and (`C`, `L`), $\texttt{RTA}^*\texttt{-FP}$ finds 8.0% and 14.9% *less* schedulable task sets than `UGB` (the latter of which is shown in Fig. 3(f)). Considering there has been discussion of superiority between global and partitioned scheduling for the sequential task model, e.g., [25], [26], [27], it is interesting to observe comparable schedulability performance between global scheduling and a generalization of partitioned scheduling for the gang task model that shares the same prioritization policy.

We also observe that the schedulability performance of $\texttt{RTA}^*\texttt{-EDF}$ is much less than that of $\texttt{RTA}^*\texttt{-FP}$. This accords with the corresponding results for the RTA framework for the sequential task model [23].

D. Comparison of Our RTA Frameworks

Finally, we present how our novel techniques in Sections IV and V and its composition (corresponding to $\texttt{RTA}^1$, $\texttt{RTA}^2$, $\texttt{RTA}^*$) improve our basic response time analysis `RTA`. Compared to `RTA` for EDF, $\texttt{RTA}^1$, $\texttt{RTA}^2$ and $\texttt{RTA}^*$ for EDF yield 7.1%, 15.7% and 22.0% overall schedulability improvement, respectively. A similar trend is observed for FP, yielding 4.3%, 9.2% and 11.7% overall schedulability improvement, respectively. In particular, if we focus on task sets with (`C`, `L`), the improvement for EDF and FP is increased to 9.0%, 25.1% and 34.0%, and 4.1%, 10.7% and 13.6%, respectively. The results demonstrate the effectiveness of the proposed two techniques and its composition in reducing pessimistic interference calculation.

VII. Conclusion

In this paper, we generalized the existing RTA framework to gang scheduling, and utilized it with existing interference calculation for the sequential task model. We then improved the RTA framework by addressing non-parallel execution constraints and over-estimation of k-interference processor occupation. As a result, the proposed RTA framework applied to FP and EDF outperforms/complements existing studies. In the future, we would like to apply the framework to other scheduling algorithms shown to be effective in the sequential model, e.g., FPZL.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2021R1A2B5B02001758, NRF-2021K2A9A1A01101570, NRF-2022R1A4A3018824).

References

[1] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky, "Nvidia a100 tensor core gpu: Performance and innovation," in *IEEE Micro 41.2*, 2021, pp. 29–35.

[2] T. Ridnik, H. Lawen, A. Noy, E. Ben Baruch, G. Sharir, and I. Friedman, "Tresnet: High performance gpu-dedicated architecture," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1400–1409.

[3] V. Young, A. Jaleel, E. Bolotin, E. Ebrahimi, D. Nellans, and O. Villa, "Combining hw/sw mechanisms to improve numa performance of multi-gpu systems," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2018, pp. 339–351.

[4] S. Kato and Y. Ishikawa, "Gang EDF scheduling of parallel task systems," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, December 2009, pp. 459–468.

[5] Z. Dong and C. Liu, "Analysis techniques for supporting hard real-time sporadic gang task systems," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, December 2017, pp. 128–138.

[6] J. Goossens and V. Berten, "Gang FTP scheduling of periodic and parallel rigid real-time tasks," *CoRR, abs/1006.2617 URL: http://arxiv.org/abs/1006.2617*, 2010.

[7] V. Berten, P. Courbin, and J. Goossens, "Gang fixed priority scheduling of periodic moldable realtime tasks," in *In 5th Junior Researcher Workshop on Real-Time Computing*, 2011, pp. 9–12.

[8] S. Collette, L. Cucu, and J. Goossens, "Integrating job parallelism in real-time scheduling theory," *Information processing letters*, vol. 106, no. 5, pp. 180–187, 2008.

[9] J. Goossens and P. Richard, "Optimal scheduling of periodic gang tasks," *Leibniz transactions on embedded systems*, pp. 04:1–04:18, 2016.

[10] S. Wasly and R. Pellizzoni, "Bundled scheduling of parallel real-time tasks," in *Proceedings of IEEE Real-Time Technology and Applications Symposium (RTAS)*, 2019, pp. 130–142.

[11] N. Ueter, M. Gunzel, G. von der Bruggen, and J.-J. Chen, "Hard real-time stationary GANG-scheduling," in *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 2021, pp. 10:1–10:19.

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

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

[14] P.-F. Dutot, G. Mounié, and D. Trystram, "Scheduling parallel tasks: Approximation algorithms, chapter 26. handbook of scheduling algorithms, models and performance analysis," 2004.

[15] Z. Dong and C. Liu, "Analysis techniques for supporting hard real-time sporadic gang task systems," *Real-Time Systems*, vol. 55, no. 4.

[16] R. Richard, J. Goossens, and S. Kato, "Comments on "gang EDF schedulability analysis"," *arXiv preprint arXiv:1705.05798*, 2014.

[17] Z. Dong and C. Liu, "Work-in-progress: Non-preemptive scheduling of sporadic gang tasks on multiprocessors," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, December 2019, pp. 512–515.

[18] G. Nelissen, J. Marcè i Igual, and M. Nasri, "Response-time analysis for non-preemptive periodic moldable gang tasks," in *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 2022, pp. 12:1–12:22.

[19] S. Lee, N. Guan, and J. Lee, "Design and timing guarantee for non-preemptive gang scheduling," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2022, to appear.

[20] W. Ali, R. Pellizzoni, and H. Yun, "Virtual gang scheduling of parallel real-time tasks," in *Proceedings of Design Automation and Test in Europe (DATE)*, 2021, pp. 270–275.

[21] A. Bhuiyan, K. Yang, S. Arefin, A. Saifullah, N. Guan, and Z. Guo, "Mixed-criticality multicore scheduling of real-time gang task systems," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, December 2019, pp. 469–480.

[22] Z. Dong, K. Yang, N. Fisher, and C. Liu, "Tardiness bounds for sporadic gang tasks under preemptive global edf scheduling," *IEEE Transactions on Parallel and Distributed Systems*, 2021, Early Access.

[23] 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. 149–158.

[24] 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.

[25] 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.

[26] T. Baker, "A comparison of global and partitioned edf schedulability tests for multiprocessors," in *International Conference on Real-Time Networks and Systems*, 2005.

[27] S. Lauzac, R. Melhem, and D. Mosse, "Comparison of global and partitioning schemes for scheduling rate monotonic tasks on a multiprocessor," in *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 1998, pp. 188–195.