Real-Time Syst (2006) 34:155–172
DOI 10.1007/s11241-006-8198-4

Approximation algorithms for scheduling real-time jobs with multiple feasible intervals

Jian-Jia Chen · Jun Wu · Chi-Sheng Shih

Published online: 31 July 2006

Abstract Time-critical jobs in many real-time applications have multiple feasible intervals. Such a job is constrained to execute from start to completion in one of its feasible intervals. A job fails if the job remains incomplete at the end of the last feasible interval. Earlier works developed an optimal off-line algorithm to schedule all the jobs in a given job set and on-line heuristics to schedule the jobs in a best effort manner. This paper is concerned with how to find a schedule in which the number of jobs completed in one of their feasible intervals is maximized. We show that the maximization problem is $\mathcal{NP}$-hard for both non-preemptible and preemptible jobs. This paper develops two approximation algorithms for non-preemptible and preemptible jobs. When jobs are non-preemptible, Algorithm Least Earliest Completion Time First (LECF) is shown to have a 2-approximation factor; when jobs are preemptible, Algorithm Least Execution Time First (LEF) is proved being a 3-approximation algorithm. We show that our analysis for the two algorithms are tight. We also evaluate our algorithms by extensive simulations. Simulation results show that Algorithms LECF and LEF not only guarantee the approximation factors but also outperform other multiple feasible interval scheduling algorithms in average.

1. Introduction

In some real-time applications, a job may have more than one feasible interval. Such a job is constrained to execute in its feasible intervals. Specifically, a job is said to complete in time if the job starts its execution in one of its feasible intervals and completes by the end of the interval. If the job remains incomplete at the end of the interval, the partial work done

J.-J. Chen (✉) · C.-S. Shih
Department of Computer Science and Information Engineering, Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan, ROC
e-mail: {r90079, cshih}@csie.ntu.edu.tw

J. Wu
Department of Information Technology, National Pingtung Institute of Commerce, Taiwan, ROC
e-mail: junwu@mail.ckitc.edu.tw

by the job is lost. The scheduler then schedules the job to execute from the start in a later feasible interval if such an interval exists. The job fails if it remains incomplete at the end of its latest feasible interval.

An example of such an application is real-time packet delivery for mobile devices in ad hoc networks. In network of this type, packets are routed and delivered by ad hoc devices. The radio signals of the devices may not always cover all the regions reachable by mobile sources and destinations. A destination in a region that is not covered by any routing device is disconnected from the network. In other words, there may be no feasible route from a source to a destination at some times. Routing devices are often resource poor and unable to buffer the packets. Consequently, packets sent in intervals when there is no feasible route are likely to be lost. A good strategy is for the source to constrain its packet transmission in time intervals during which there are feasible routes to the destination. These time intervals are called feasible intervals in the previous paragraph. We confine our attention here to networks where the movement of the devices are predictable. Hence, the starts and ends of all feasible intervals can be determined with sufficient accuracy at the times when packets are buffered for transmission and routes to the destination are discovered. According to our model, the transmission of each real-time packet is a job with a deadline, which is a time instance by which the job must complete (i.e., the packet reaches the destination). The transmission job has multiple feasible intervals in general and must start and complete within one of its feasible intervals.

The optional jobs in the error-cumulative imprecise computation model studied in Cheong (1992), Liu et al. (1991), Shih et al. (1991), Shih and Liu (1995) are also examples of jobs with multiple feasible intervals. According to the imprecise computation model, a job consists of two parts: mandatory part and optional part. The mandatory part must complete by its deadline, and the optional part can be skipped if there are not enough resources. Skipping the optional part introduces error into the result produced by the job. In some real-time applications (e.g., radar tracking) the error of a periodic task is the cumulated error produced by incomplete optional parts of jobs in the task. The error-cumulative model introduces a threshold for the cumulative error of each task. When the cumulative error becomes greater than the threshold, the task fails. (In a tracking system, the system may lose track of the target if the cumulative error becomes greater than the given threshold.) To ensure that the error is under the threshold, the optional part of a job in one of N periods must be executed to completion by the end of its period, where N is derived from the error threshold of the task. The other optional jobs in the N periods can be discarded entirely. We can view the optional parts of the jobs in N periods as a job with N feasible intervals, which are the time intervals left after the mandatory parts of the jobs complete. As long as the job with N feasible intervals completes in time, the error of the periodic task is under the allowed threshold.

Shih et al. (2003) showed that it is $\mathcal{NP}$-complete to determine if there exists a schedule for a set of jobs such that all the jobs meet their timing constraints. They developed optimal and sub-optimal algorithms for off-line scheduling and a family of heuristics for on-line scheduling. However, the goals of the algorithms are to complete all multiple feasible interval jobs in the job set. In this paper, we consider the systems in which it is acceptable not to complete all the multiple feasible interval jobs in the job set in time. Examples of such systems are the skip-over model (Koren and Shasha, 1995), reward-based model (Aydin et al., 1999), (error-cumulative) imprecise computation model (Liu et al., 1991; Shih et al., 1991; Shih and Liu, 1995), and (m,k)-rm guarantee model (Quan and Hu, 2000; Hamdaoui and Ramanathan, 1995). This paper is concerned with how to find a schedule which maximizes the number of jobs completed in one of their feasible intervals. When jobs are preemptible

and each job has only one feasible interval, the optimization problem was studied in Baptiste (1999), Lawler (1990). In this paper, we develop two algorithms providing worst-case guarantees. Algorithm Least Earliest Completion Time First (LECF) schedules non-preemptible multiple feasible interval jobs. Theoretical analysis shows that Algorithm LECF is with a 2-approximation factor, in which the number of multiple feasible interval jobs completed in time in the derived schedule is at least one half of that in any non-preemptive schedule. (We will define the approximation factor later.) On the other hand, Algorithm Least Execution Time First (LEF) is designed for the case that jobs are preemptible. The approximation factor for Algorithm LEF is shown being 3. We also show the tightness of our analysis on the performance guarantees by providing a set of input instances. In addition, we evaluate our algorithms by extensive simulations. Simulation results show that Algorithms LECF and LEF not only guarantee the approximation factors but also outperform other existing algorithms.

The rest of this paper is organized as follows: Section 2 describes the formal model and defines the scheduling problems for jobs with multiple feasible intervals. Section 3 presents a 2-approximation algorithm for non-preemptible multiple feasible interval jobs. In succession, Section 4 gives a 3-approximation algorithm for preemptible multiple feasible interval jobs. Section 5 presents the evaluation results for the developed algorithms against existing approaches. Section 6 concludes this paper.

2. Formal models and problem statements

Throughout this paper, the term *job* refers to an instance of computation, or the transmission of a data packet, or the retrieval of a file, and so on. A *task* is a sequence of jobs that have identical or similar characteristics and timing requirements (Liu and Layland, 1973; Han and Lin, 1992; B. Sprunt et al., 1989). We focus on jobs regardless of the types of tasks to which they belong and call jobs J_1, J_2, and so on.

2.1. Multiple feasible interval jobs

Each multiple feasible interval job is characterized by its execution time and a set of feasible intervals. The *execution time* of a job, denoted by e, is the amount of time required to complete the job when it executes alone and has all the resources it requires. Throughout our discussion, we assume that for the purpose of determining whether each job can complete by its deadline, knowing its worst case execution time (WCET) is sufficient. By execution time of a job, we mean its WCET. Associated with each job is a set of disjointed time intervals, called *feasible interval*s. The job can execute only in its feasible intervals. A job is *ready* to execute at time t if it has not completed at time t and time t is in one of its feasible intervals. Once a job begins to execute in a feasible interval, it must complete by the end of the interval in order to produce a correct result. The partial work done by the job is lost if the job remains incomplete at the end of the interval. In that case, the scheduler might reschedule the job to execute from the start in a later feasible interval of the job if such an interval exists.

We denote a feasible interval by $I = (L, R]$ where L and R are non-negative rational numbers which represent the start time and end time of the interval, respectively. $I_{i,j}$ denotes the j-th feasible interval of multiple interval job J_i, where $L_{i,j}$ and $R_{i,j}$ denote the start time and end time of $I_{i,j}$, respectively. The set of feasible intervals of job J_i is denoted by $\mathbb{I}_i = \{I_{i,1}, I_{i,2}, \ldots, I_{i,n(i)}\}$, where $n(i)$ is the number of feasible intervals for job J_i, and the feasible intervals in the set are indexed in an ascending order of their start times. We

represent a multiple feasible interval job J_i by $J_i = (e_i, \mathbb{I}_i)$. Hereafter, we focus on this kind of jobs and omit "multiple feasible interval" as long as there is no ambiguity.

According to the traditional definition, a job meets its timing constraint (or the job is timely) if it completes by its deadline. This definition of timeliness needs to be generalized when jobs have multiple feasible intervals. We call a failed attempt to complete the execution of a job in one of its feasible intervals a *deadline miss*. More precisely, a deadline miss occurs when a job executing in a feasible interval remains incomplete at the end of the feasible interval. To meet the timing constraint, the job has to complete its execution within any of its feasible intervals. When a job misses the deadline for its last feasible interval, the job fails to meet its timing constraint. The following definition states the timing constraint of a job.

Definition 1 (In-Time Completion). An execution of a job J completes in time if and only if there is no deadline miss since it starts and before it completes. A job J meets its timing constraint, or simply that it completes in time, if and only if one of its execution completes in time.

A *schedule* for a job set $\mathbb{J}$, denoted by $S_{\mathbb{J}}$, is the time sequence of executing jobs in the job set. For a given schedule, the *start time* and *completion time* of job J_i, denoted by s_i and c_i, are the time instants at which job J_i starts and completes its execution, respectively. When jobs are preemptible, the scheduler may interrupt the execution of the jobs. Hence, a preemptible job may execute in several non-consecutive time intervals in a schedule. When a job is non-preemptible, the job must execute from start to completion in one consecutive time interval. If there is no interrupt during the execution of a job in a schedule, the job is *non-interrupted* in the schedule; otherwise, the job is *preempted* or *interrupted* in the schedule. Since removing the executions of the jobs that do not complete in time does not affect schedulability of the completed jobs in any schedule, we only focus our discussions on schedules that complete all the executed jobs in time. A schedule is said *feasible* when all the jobs in the job set complete in time as defined in Definition 1. The following theorem stated in Shih et al. (2003) shows that unless $\mathcal{P} = \mathcal{NP}$, there does not exist any polynomial-time algorithm to determine if there exists a feasible schedule for all the jobs in job set $\mathbb{J}$.

Theorem 1. [*NP-Hardness for Multiple Interval Job Scheduling Problem* (*Shih et al., 2003*)] *Finding a schedule to complete all the jobs in a set of interval jobs in time is $\mathcal{NP}$-complete.*

2.2. Problem formulation

This paper is concerned with an optimization problem to derive a schedule such that the number of multiple feasible interval jobs completed in time is maximized. Both non-preemptible and preemptible jobs are considered in this paper. We state the optimization problems in the following.

1. Non-preemptible Interval Job Scheduling (n-IJS) Problem: We are given a job set $\mathbb{J} = \{J_1, J_2, \ldots, J_M\}$ of non-preemptible multiple feasible interval jobs. The objective is to find a feasible schedule for a subset of job set $\mathbb{J}$ such that the size of the subset is maximized.

2. Preemptible Interval Job Scheduling (IJS) Problem: We are given a job set $\mathbb{J} = \{J_1, J_2, \ldots, J_M\}$ of preemptible multiple feasible interval jobs. The objective is to find a feasible schedule for a subset of job set $\mathbb{J}$ such that the size of the subset is maximized.

We assume that all the jobs in the job set $\mathbb{J}$ are equally important in this paper. When the jobs have different importance levels, the developed algorithms can be applied for each importance level one at a time. The following corollary is a simple extension from Theorem 1.

Corollary 1. *Both of the IJS and n-IJS problems are $\mathcal{NP}$-hard.*

Instead of finding an optimal solution for the IJS and *n*-IJS problems, we focus on developing polynomial-time algorithms which find approximated solutions with worst-case guarantees for the two optimization problems. An algorithm for a maximization problem is with an *α-approximation factor* (ratio) (Vazirani, 2001, section 1) if its derived solution is no less than $\frac{1}{\alpha}$ times of the optimal solution. Hence, an α-approximation algorithm for the IJS or n-IJS problem guarantees that the number of jobs completed in time in its derived schedule is no less than $\frac{1}{\alpha}$ times of that in an optimal schedule.

3. Algorithm least earliest completion time first (LECF) for non-preemptible jobs

In this section, an approximation algorithm is proposed for the n-IJS problem. The rationale of this algorithm is to always schedule the job which is able to complete in time earliest among the ready jobs in a greedy manner. We show that such an algorithm has a 2-approximation factor and that our analysis is tight by providing a set of input instances by the end of this section.

Before we present the algorithm, we define several terms used in this section. *Scheduling time* is the time instant at which the scheduler selects one ready job to execute. For the sake of simplicity, when defining the following terms for job J_i, we assume that there is only one job J_i in the system. The *earliest completion interval at scheduling time t*, denoted as $I_{i,k}$, for job J_i is the first feasible interval in which job J_i completes in time and its start time is no earlier than t. In other words, $I_{i,k}$ is the feasible interval whose start time is the minimum and both $R_{i,k} - e_i \geq t$ and $R_{i,k} - L_{i,k} \geq e_i$ hold. The *earliest completion time* of job J_i at scheduling time t is the least time instant at which J_i completes in time when it starts its execution no earlier than t. When scheduling time t is in the earliest completion interval $I_{i,k}$ of job J_i, J_i can start its execution at time instant t; otherwise, job J_i can start at the start time of feasible interval $I_{i,k}$, i.e., $L_{i,k}$. Thus, the earliest completion time of job J_i at scheduling time t is the sum of the maximum value of t and the start time of its earliest completion interval $I_{i,k}$ at time t, and its execution time, i.e., $\max\{t, L_{i,k}\} + e_i$. A job J_i is said to be *unschedulable at scheduling time t* if there is no earliest completion interval at scheduling time t.

Algorithm 1. Least Earliest Completion Time First (LECF) Algorithm

Input $\mathbb{J}$*;*
Output *A feasible schedule for a subset $\mathbb{J}'$ of $\mathbb{J}$*;
1: *remove feasible intervals $I_{i,j}$ from $\mathbb{I}_i$ with $e_i > R_{i,j} - L_{i,j}$ for every job $J_i \in \mathbb{J}$*;
2: $t \leftarrow 0$; $\mathbb{J}' \leftarrow \emptyset$;
3: *set $I_{i,1}$ as the earliest completion interval at scheduling time* 0 *for every job $J_i \in \mathbb{J}$*;

4: **repeat**
5: $J_{i^*} \leftarrow$ *the job with the least earliest completion time at* t *in* $\mathbb{J} \setminus \mathbb{J}'$;
6: $\mathbb{J}' \leftarrow \mathbb{J}' \cup \{J_{i^*}\}$;
7: *schedule* J_{i^*} *in* $(\max\{t, L_{i^*,k}\}, \max\{t, L_{i^*,k}\} + e_{i^*}]$;
8: $t \leftarrow \max\{t, L_{i^*,k}\} + e_{i^*}$;
9: *update the earliest completion interval at* t *for every job in* $\mathbb{J} \setminus \mathbb{J}'$;
//set $I_{i,k}$ *as the earliest completion interval at* t *for* J_i *such that* k *is the minimum* j *with* $R_{i,j} - e_i \geq t$;
10: **until** *all the jobs in* $\mathbb{J} \setminus \mathbb{J}'$ *are unschedulable at* t *or* $\mathbb{J} \setminus \mathbb{J}' = \emptyset$;
11: *return the schedule of* $\mathbb{J}'$;

The Least Earliest Completion First (LECF) strategy is adopted for the n-IJS problem. When Algorithm LECF, presented in Algorithm 1, starts, scheduling time t is initialized as the minimal of the start time of the first feasible interval of all the jobs in job set $\mathbb{J}$. Algorithm LECF repeats the following steps to select a job and advance scheduling time t. At scheduling time t, we greedily select job J_{i^*} whose earliest completion time is the least among that of all the unselected jobs. Job J_{i^*} starts to execute at the maximum value of t and the start time of its earliest completion interval $I_{i^*,k}$ and completes at its earliest completion time at time t. In other words, job J_{i^*} is scheduled in interval $(\max\{t, L_{i^*,k}\}, \max\{t, L_{i^*,k}\} + e_{i^*}]$, where $L_{i^*,k}$ is the start time of its earliest completion interval at t. Then, scheduling time t is advanced to the (earliest) completion time of the job J_{i^*} at t, i.e., $t = \max\{t, L_{i^*,k}\} + e_{i^*}$. Algorithm LECF terminates when all the unselected jobs are unschedulable, or one interval is selected for every job in job set $\mathbb{J}$ and returns the derived schedule.

The time complexity of this algorithm is $O(|\mathbb{J}|^2 + \Sigma_{J_j \in \mathbb{J}} |\mathbb{I}_j|)$, dominated by updating the earliest completion intervals at scheduling time t at Line 9 in Algorithm 1 by an incremental manner. (Note that $|\mathbb{X}|$ denotes the cardinality of any set $\mathbb{X}$ in this paper.) For the rest of this section, let $\mathbb{J}^{\text{LECF}}$ be the set of jobs scheduled by Algorithm LECF. Clearly, all the jobs in $\mathbb{J}^{\text{LECF}}$ complete in time. The optimality, i.e., the approximation factor, of Algorithm LECF is shown as follows.

Theorem 2. *Algorithm* LECF *is a 2-approximation algorithm for the n-IJS problem.*

Proof: Let $\mathbb{J}^*$ be a subset of $\mathbb{J}$, including an optimal subset, such that there is a feasible schedule S^* for job set $\mathbb{J}^*$. It suffices to prove this theorem by showing that the size of job set $\mathbb{J}^{\text{LECF}}$ by Algorithm LECF is at least one half of the size of job set $\mathbb{J}^*$.

Let $\mathbb{J}_1^*$ and $\mathbb{J}_2^*$ be the intersection of $\mathbb{J}^{\text{LECF}}$ and $\mathbb{J}^*$, i.e., $\mathbb{J}_1^* \equiv \mathbb{J}^{\text{LECF}} \cap \mathbb{J}^*$, and the relative complement of $\mathbb{J}^*$ in $\mathbb{J}_1^*$, i.e., $\mathbb{J}_2^* \equiv \mathbb{J}^* \setminus \mathbb{J}_1^*$, respectively. In other words, $\mathbb{J}_1^*$ consists of the jobs both in $\mathbb{J}^*$ and $\mathbb{J}^{\text{LECF}}$, while $\mathbb{J}_2^*$ consists of the jobs that are in $\mathbb{J}^*$ but not in $\mathbb{J}^{\text{LECF}}$. We will show that $|\mathbb{J}_1^*| \leq |\mathbb{J}^{\text{LECF}}|$ and $|\mathbb{J}_2^*| \leq |\mathbb{J}^{\text{LECF}}|$. When the above two conditions hold, we will have

$$|\mathbb{J}^{\text{LECF}}| \geq 0.5\big(|\mathbb{J}_1^*| + |\mathbb{J}_2^*|\big) = 0.5|\mathbb{J}^*|. \tag{1}$$

Naturally, condition $|\mathbb{J}_1^*| \leq |\mathbb{J}^{\text{LECF}}|$ holds. Hence, it remains to show that condition $|\mathbb{J}_2^*| \leq |\mathbb{J}^{\text{LECF}}|$ holds.

We divide schedule S^* into $|\mathbb{J}^{\text{LECF}}|$ time intervals. Let c_j be the completion time for j-th selected job in the derived schedule of Algorithm LECF for $1 \leq j \leq |\mathbb{J}^{\text{LECF}}|$ and c_0 be 0. That is, in the j-th iteration of Algorithm LECF, scheduling time t is c_{j-1} before t is updated (Line 8). For the derived schedule of $\mathbb{J}^{\text{LECF}}$, there is only one job executed in each of the time

intervals $(c_{j-1}, c_j]$ for $j = 1$ to $|\mathbb{J}^{\text{LECF}}|$. Let S_2^* be the schedule by removing the executions of the jobs in $\mathbb{J}_1^*$ from schedule S^*. Hence, S_2^* is a feasible schedule of $\mathbb{J}_2^*$. Note that the jobs in S_2^* are executed one by one because all the jobs are non-preemptible.

We claim that, in S_2^*, there is at most one job whose start time is in interval $[c_{j-1}, c_j)$ for $j = 1$ to $|\mathbb{J}^{\text{LECF}}|$. We prove this statement by contradiction. Assume that, in schedule S_2^*, there exists an interval $[c_{j-1}, c_j)$ for some $1 \le j \le |\mathbb{J}^{\text{LECF}}|$ such that at least two jobs in job set $\mathbb{J}_2^*$ start their executions in this time interval. Let J_i be the job whose start time s_i is the earliest among that for jobs scheduled in this interval. Since, in schedule S_2^*, there is another job started before time c_j, we know that the completion time for job J_i, i.e., $s_i + e_i$, in S_2^* must be less than c_j. Therefore, we have $c_{j-1} \le s_i$ and $s_i + e_i < c_j$. That is, the earliest completion time of job J_i is less than that of the job scheduled in interval $[c_{j-1}, c_j)$ in schedule S^{LECF}. According to Algorithm LECF, job J_i must be selected to execute at scheduling time c_{j-1}. This contradicts the least earliest completion first strategy. Hence, there is at most one job whose start time in S_2^* is in $[c_{j-1}, c_j)$ for $j = 1$ to $|\mathbb{J}^{\text{LECF}}|$.

Furthermore, in job set $\mathbb{J}_2^*$, there is no job which starts its execution in $[c_{|\mathbb{J}^{\text{LECF}}|}, \infty]$ in S_2^*. This is because it contradicts the termination condition of Algorithm LECF: all the jobs in $\mathbb{J} \setminus \mathbb{J}^{\text{LECF}}$ are unschedulable at scheduling time $c_{|\mathbb{J}^{\text{LECF}}|}$. Hence, there are at most $|\mathbb{J}^{\text{LECF}}|$ jobs starting their executions in S_2^*. Therefore, we know $|\mathbb{J}_2^*| \le |\mathbb{J}^{\text{LECF}}|$. By Eq. (1), Algorithm LECF has a 2-approximation factor. □

We conclude this section by showing that our analysis in Theorem 2 is tight. Consider the input instance $\mathbb{J} = \{J_1, J_2\}$ as illustrated in Fig. 1(a), where $J_1 = (b, \{(0, b], (b + \epsilon, 2b + \epsilon]\})$ and $J_2 = (b + \epsilon, \{(0, b + \epsilon]\})$ ($b > 0$ and $\epsilon > 0$). The optimal schedule is to execute J_1 in its second feasible interval and J_2 in its first feasible interval (refer to Fig. 1(b)). Algorithm LECF schedules only job J_1 in its first feasible interval and drops job J_2 (refer to Fig. 1(c)).

4. Algorithm least execution time first (LECF) for preemptible jobs

In this section, we are concerned with the preemptible interval job scheduling (IJS) problem. We develop Least Execution Time First (LEF) Algorithm with a constant approximation factor based on the well-known earliest-deadline-first (EDF) scheduling strategy (Baptiste, 1999; Liu, 2000). The algorithm will be shown being a 3-approximation algorithm followed by a set of tight input instances.

Intuitively, Algorithm LECF could also be adopted for the IJS problem because a feasible schedule produced by non-preemptive scheduling algorithm is also feasible for preemptible jobs. However, it is not difficult to show that the approximation factor of Algorithm LECF for the IJS problem is $\Omega(|\mathbb{J}|)$. Consider the input instance: $\mathbb{J} = \{J_1, J_2, \ldots, J_N\}$, where $J_i = (2^{i-1}, \{(2^{N-1} - 1 - \Sigma_{j=2}^{i} 2^{j-2}, 2^{N-1} - 1 + 2^{i-1}]\})$ for $i = 1, \ldots, N$. For such an input instance, Algorithm LECF will choose only one job J_{i^*} to execute in $(2^{N-1} - 1 - \Sigma_{j=2}^{i^*} 2^{j-2}, 2^{N-1}]$, since the earliest completion time of these N jobs is the same, i.e., 2^{N-1}.

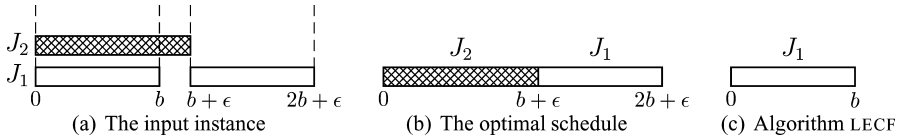


Fig. 1 A tight example for Algorithm LECF

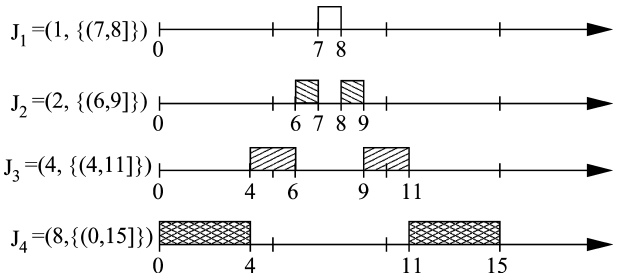


Fig. 2 Feasible schedule for preemptible interval job set $\mathbb{J}$

All the other jobs are unschedulable after time instant 2^{N-1}. However, another algorithm may schedule job J_i in $(2^{N-1} - 1 - \Sigma_{j=2}^{i} 2^{j-2}, 2^{N-1} - 1 - \Sigma_{j=2}^{i} 2^{j-2} + \max\{1, 2^{i-2}\}]$ and $(2^{N-1} - 1 + \max\{1, 2^{i-1} - 2^{i-2}\}, 2^{N-1} - 1 + 2^{i-1}]$. Such a schedule is feasible for job set $\mathbb{J}$. For example, when $N = 4$ in the above input instances, $J_1 = (1, \{(7, 8]\})$, $J_2 = (2, \{(6, 9]\})$, $J_3 = (4, \{(4, 11]\})$, and $J_4 = (8, \{(0, 15]\})$. Algorithm LECF only schedules one of these four jobs to complete at time instant 8. However, another scheduling algorithm may schedule all four jobs to complete in time. Figure 2 illustrates the schedule.

In this schedule, job J_1 executes to completion in its feasinle interval $(7, 8]$. Job J_2, J_3, and J_4 are diveded into two parts to execute in their feasible intervals. Therefore, the approximation factor of Algorithm LECF for the IJS problem is $\Omega(|\mathbb{J}|)$.

Before we present the algorithm, we again define the terms used in this section. Similar to the algorithms proposed by Shih et al. (2003), a solution for the IJS problem has two parts: feasible interval selection and job scheduling. The first part selects one feasible interval $I_{i,j}$ for job J_i if a feasible schedule exists. Job J_i with selected feasible interval $I_{i,j}$ is denoted by $\hat{J}_i$. Naturally, job $\hat{J}_i$ can be treated as a traditional real-time job: the arrival time and deadline for job $\hat{J}_i$ are the start time and end time of its selected feasible interval, respectively. Suppose that $\mathbb{J}^\dagger$ is the set of the jobs selected to execute and $\hat{\mathbb{J}}^\dagger$ is the job set consisting of jobs with selected feasible intervals in job set $\mathbb{J}^\dagger$. The schedulability of job set $\hat{\mathbb{J}}^\dagger$ can be conducted by applying the traditional earliest-deadline-first (EDF) strategy: whenever no job is executed, and one job in job set $\hat{\mathbb{J}}^\dagger$ is ready and not completed yet, schedule the ready job with the earliest deadline. The optimality of the EDF strategy guarantees that there exists a feasible schedule for job set $\hat{\mathbb{J}}^\dagger$ if and only if the schedule derived by the EDF strategy is feasible (Carlier, 1982). For the rest of this paper, we denote the EDF schedule for a job set $\mathbb{J}$ as $S_{\mathbb{J}}^{\text{EDF}}$. A schedulable subset of job set $\mathbb{J}$, including the optimal subset, is denoted by $\mathbb{J}^*$. The set of jobs selected by Algorithm LEF is denoted by $\mathbb{J}^{\text{LEF}}$. The intersection of $\mathbb{J}^{\text{LEF}}$ and $\mathbb{J}^*$ is denoted by $\mathbb{J}_1^*$, i.e., $\mathbb{J}_1^* \equiv \mathbb{J}^{\text{LEF}} \cap \mathbb{J}^*$, and the relative complement of $\mathbb{J}^*$ in $\mathbb{J}_1^*$ is denoted by $\mathbb{J}_2^*$, i.e., $\mathbb{J}_2^* \equiv \mathbb{J}^* \setminus \mathbb{J}_1^*$. In other words, job set $\mathbb{J}_1^*$ consists of the jobs both in $\mathbb{J}^*$ and $\mathbb{J}^{\text{LEF}}$, while job set $\mathbb{J}_2^*$ consists of the jobs that are in $\mathbb{J}^*$ but not in $\mathbb{J}^{\text{LEF}}$.

The challenge of the IJS problem is two-fold: which job in job set $\mathbb{J}$ should be scheduled at scheduling time t and which feasible interval for each selected job should be selected. Algorithm Least Execution Time First (LEF) selects the job in a non-descending order of execution time of jobs of the input job set $\mathbb{J}$. Initially, $\mathbb{J}^{\text{LEF}}$ and $\hat{\mathbb{J}}^{\text{LEF}}$ are both empty sets. The working copy $\mathbb{T}$ is equal to $\mathbb{J}$ when the algorithm starts. In each iteration, the algorithm selects one job whose execution time is the least among that for all the jobs in job set $\mathbb{T}$, denoted by J_{i^*}. If there exists a feasible interval $I_{i^*,j}$ of job J_{i^*} such that the job set consisting of jobs in $\hat{\mathbb{J}}^{\text{LEF}}$ and job J_{i^*} are schedulable, then job J_{i^*} is inserted into job set $\mathbb{J}^{\text{LEF}}$ and job

Algorithm 2 : Least Execution Time First (LEF)

Input: $\mathbb{J}$;
Output: A feasible schedule for a subset $\mathbb{J}^{\mathrm{LEF}}$ of $\mathbb{J}$;
1: $\mathbb{T} \leftarrow \mathbb{J}$; $\mathbb{J}^{\mathrm{LEF}} \leftarrow \emptyset$; $\hat{\mathbb{J}}^{\mathrm{LEF}} \leftarrow \emptyset$;
2: **while** $\mathbb{T} \neq \emptyset$ **do**
3: let $J_{i^*} \leftarrow \arg_{J_i \in \mathbb{T}} \min\{e_i\}$;
4: **for each** $I_{i^*,j} \in \mathbb{I}_{i^*}$ **do**
5: transform J_{i^*} by selecting interval $I_{i^*,j}$ as $\hat{J}_{i^*}$;
6: **if** $\{\hat{J}_{i^*}\} \cup \hat{\mathbb{J}}^{\mathrm{LEF}}$ is schedulable by EDF **then**
7: $\mathbb{J}^{\mathrm{LEF}} \leftarrow \mathbb{J}^{\mathrm{LEF}} \cup \{J_{i^*}\}$; $\hat{\mathbb{J}}^{\mathrm{LEF}} \leftarrow \hat{\mathbb{J}}^{\mathrm{LEF}} \cup \{\hat{J}_{i^*}\}$;
8: break;
9: $\mathbb{T} \leftarrow \mathbb{T} \setminus \{J_{i^*}\}$;
10: return $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$;

$\hat{J}_{i^*}$ is inserted into job set $\hat{\mathbb{J}}^{\mathrm{LEF}}$; otherwise, job J_{i^*} is not selected. (We break ties arbitrarily.) After removing job J_{i^*} from job set $\mathbb{T}$, the algorithm repeats the iteration until job set $\mathbb{T}$ becomes an empty set. The pseudo codes for Algorithm LEF are shown in Algorithm **??**. The schedule derived from Algorithm LEF completes all the jobs in $\mathbb{J}^{\mathrm{LEF}}$ in time. Because the EDF schedulability test could be done in $O(|\mathbb{J}|)$ if the jobs are sorted according to their deadlines *a priori*, the time complexity of this algorithm is $O(|\mathbb{J}| \Sigma_{J_i \in \mathbb{J}} |\mathbb{I}_i|)$.

In the following, we show the approximation factor of Algorithm LEF. We will use the following properties for EDF schedules to prove the approximation factor of Algorithm LEF, where (P2) comes directly from (P1) and (P3) is due to the greedy strategy of EDF schedules:

(P1). A set $\hat{\mathbb{J}}$ of jobs, each of which has one selected feasible interval, is schedulable by the EDF strategy if and only if there exists a schedule to complete all the jobs in $\hat{\mathbb{J}}$ in time (Carlier, 1982).
(P2). If a set $\hat{\mathbb{J}}$ of jobs, each of which has one selected feasible interval, is schedulable by the EDF strategy, any subset of $\hat{\mathbb{J}}$ is also schedulable by the EDF strategy.
(P3). During the time interval from the start time to the completion time of a job in the EDF schedule, there is at least one non-interrupted job.

The 3-approximation factor can be proved by showing that the cardinality of job set $\hat{\mathbb{J}}^*_2$ is at most twice of that of job set $\hat{\mathbb{J}}^{\mathrm{LEF}}$. This is because the cardinality of job set $\mathbb{J}^*_1$ is no greater than that of $\mathbb{J}^{\mathrm{LEF}}$. The following lemmas state two important properties for job sets $\hat{\mathbb{J}}^{\mathrm{LEF}}$ and $\hat{\mathbb{J}}^*_2$.

Lemma 1. *The job set consisting of one job $\hat{J}_h$ in job set $\hat{\mathbb{J}}^*_2$, and all the jobs in job set $\hat{\mathbb{J}}^{\mathrm{LEF}}$ each of whose execution time is no greater than that of $\hat{J}_h$, i.e., $\{\hat{J}_h\} \cup \{\hat{J}_j \mid \hat{J}_j \in \hat{\mathbb{J}}^{\mathrm{LEF}}$ and $e_j \leq e_h\}$, is not schedulable.*

Proof: This lemma can be proved by contradiction.
Suppose that job set $\hat{\mathbb{J}}_h \equiv \{\hat{J}_h\} \cup \{\hat{J}_j \mid \hat{J}_j \in \hat{\mathbb{J}}^{\mathrm{LEF}}$ and $e_j \leq e_h\}$ is schedulable for job $\hat{J}_h$ in $\hat{\mathbb{J}}^*_2$. The job set $\hat{\mathbb{J}}^{\mathrm{LEF}_h}$ is defined to be the job set that consists of all the jobs in job set $\hat{\mathbb{J}}^{\mathrm{LEF}}$ before Algorithm LEF selects job J_h. Because job set $\hat{\mathbb{J}}_h$ is schedulable and $\hat{\mathbb{J}}^{\mathrm{LEF}_h} \cup \{\hat{J}_h\}$ is a subset of $\hat{\mathbb{J}}_h$, we know that job set $\hat{\mathbb{J}}^{\mathrm{LEF}}_h \cup \{\hat{J}_h\}$ is schedulable based on the EDF properties

(P1) and (P2). In other words, Algorithm LEF would insert job J_h into job set $\mathbb{J}^{\text{LEF}}$. However, it contradicts with the assumption that job J_h is in job set $\mathbb{J}_2^*$ and not in job set $\mathbb{J}^{\text{LEF}}$. We reach the contradiction. □

Lemma 2. *Suppose that $\hat{J}_j$ is a non-interrupted job which is scheduled in $S^{\text{EDF}}_{\hat{\mathbb{J}}^{\text{LEF}}}$ and whose execution time is the least. If job set $\hat{\mathbb{J}}_2^*$ is not empty, the execution time of any job in $\hat{\mathbb{J}}_2^*$ is no less than that of $\hat{J}_j$.*

Proof: By the EDF property (P3), we know that such a job $\hat{J}_j$ exists. We prove this lemma by contradiction. Assume that $\hat{J}_h$ is a job in job set $\hat{\mathbb{J}}_2^*$ and its execution time is less than that of $\hat{J}_j$. We will show that job J_h should be selected by Algorithm LEF. Let L_h be the start time of the selected feasible interval of $\hat{J}_h$ in $S^{\text{EDF}}_{\hat{\mathbb{J}}_2^*}$. By Lemma 1, there must be at least one job executed in the time interval $(L_h, L_h + e_h]$ in schedule $S^{\text{EDF}}_{\hat{\mathbb{J}}^{\text{LEF}}}$.

In the following, we consider two cases: (1) there is only one non-interrupted job in schedule $S^{\text{EDF}}_{\hat{\mathbb{J}}^{\text{LEF}}}$ and (2) there are more than one non-interrupted job in schedule $S^{\text{EDF}}_{\hat{\mathbb{J}}^{\text{LEF}}}$.

Case (1): By the EDF property (P3), all of the jobs executed after L_h must be preempted by the non-interrupted job $\hat{J}_j$ in schedule $S^{\text{EDF}}_{\hat{\mathbb{J}}^{\text{LEF}}}$. In other words, all of the jobs in job set $\hat{\mathbb{J}}^{\text{LEF}}$, excluding job $\hat{J}_j$, start their executions before the start time s_j of $\hat{J}_j$, and complete their executions after the completion time of $\hat{J}_j$, i.e., $s_j + e_j$. If $s_j \geq L_h$, we could *postpone* all of the execution pieces of the preempted jobs executed in time interval $(L_h, L_h + e_h]$ in schedule $S^{\text{EDF}}_{\hat{\mathbb{J}}^{\text{LEF}}}$ to time interval $(s_j, s_j + e_j]$. Figure 3(a) illustrates such a schedule before the transformation and Fig. 3(b) illustrates the feasible schedule after the transformation.

Similarly, If $s_j < L_h$, we could *advance* all of the execution pieces of the preempted jobs executed in time interval $(L_h, L_h + e_h]$ in schedule $S^{\text{EDF}}_{\hat{\mathbb{J}}^{\text{LEF}}}$ to time interval $(s_j, s_j + e_j]$. Figure 3(c) illustrates such a schedule before the transformation and Fig. 3(d) illustrates the corresponding feasible schedule after the transformation.

Since $e_j > e_h$, we know that job set $\hat{\mathbb{J}}^{\text{LEF}} \setminus \{\hat{J}_j\}$ is schedulable even when no job is executed in time interval $(L_h, L_h + e_h]$. Therefore, we could know that job set $\hat{\mathbb{J}}^{\text{LEF}} \setminus \{\hat{J}_j\} \cup \{\hat{J}_h\}$ is schedulable by scheduling job $\hat{J}_h$ in time interval $(L_h, L_h + e_h]$. By the EDF properties (P1) and (P2), we know that job set $\{\hat{J}_i \mid \hat{J}_i \in \hat{\mathbb{J}}^{\text{LEF}} \text{ and } e_i \leq e_h\} \cup \{\hat{J}_h\}$ is also schedulable. Algorithm LEF should insert $\hat{J}_h$ into $\hat{\mathbb{J}}^{\text{LEF}}$, which contradicts our assumption that $\hat{J}_h$ is not in $\hat{\mathbb{J}}^{\text{LEF}}$.

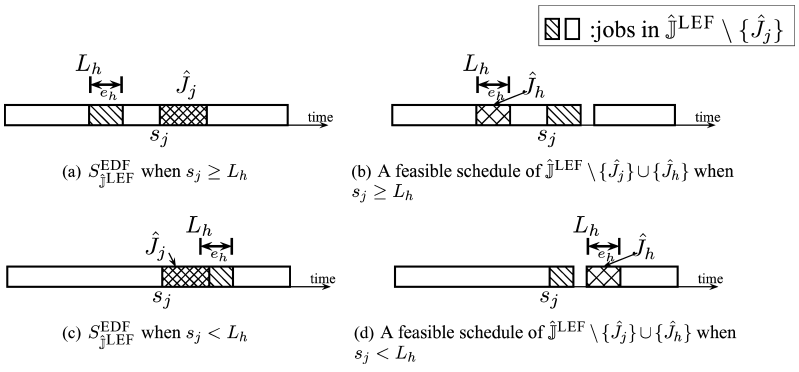
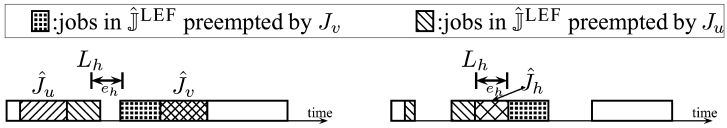


Fig. 3 An illustrative example for the proof in case (1) in Lemma 2



(a) No non-interrupted job in $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$ in $(L_h, L_h+e_h]$ (b) A feasible schedule of $\hat{\mathbb{J}}^{\mathrm{LEF}} \setminus \{\hat{J}_u, \hat{J}_v\} \cup \{\hat{J}_h\}$

Fig. 4 An illustrative example for the second case of the proof for Lemma 2

Case (2): We consider the case that there are more than one non-interrupted job in schedule $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$. The proof for the case when no non-interrupted job starts executions before L_h or completes executions after L_h in schedule $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$ is similar to that for Case (1). Hence, we only prove the other two sub-cases for this lemma:

Case (2.1): No non-interrupted job is executed in time interval $(L_h, L_h + e_h]$ in schedule $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$:

Figures 4(a) and 4(b) illustrate an example for this subcase. Let $\hat{J}_u$ be the last non-interrupted job starting before L_h and job $\hat{J}_v$ be the first non-interrupted job starting after $L_h + e_h$, where s_u and s_v denote the start times of $\hat{J}_u$ and $\hat{J}_v$, respectively. We know that all of the jobs executed in time interval $(L_h, L_h + e_h]$ if there is any are preempted either by job $\hat{J}_u$ or job $\hat{J}_v$ in schedule $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$. For the jobs which are preempted by job $\hat{J}_u$, we can advance all of their execution pieces in time interval $(L_h, L_h + e_h]$ in schedule $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$ to time interval $(s_u, s_u + e_u]$. On the other hand, for those jobs preempted by job $\hat{J}_v$, we can postpone all of their execution pieces in time interval $(L_h, L_h + e_h]$ in schedule $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$ to time interval $(s_v, s_v + e_v]$. After that, we could schedule job $\hat{J}_h$ in time interval $(L_h, L_h + e_h]$. Since both e_u and e_v are greater than e_h, the revised schedule could complete all the jobs in job set $\hat{\mathbb{J}}^{\mathrm{LEF}} \setminus \{\hat{J}_u, \hat{J}_v\} \cup \{\hat{J}_h\}$ in time. Similar to the proof for Case (1), job J_h would be selected for execution in Algorithm LEF.

Case (2.2): More than one non-interrupted job are executed in time interval $(L_h, L_h + e_h]$ in schedule $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$:

For the second sub-case, the proof is very similar. Since $e_h < e_u$ and $e_h < e_v$, we know that there are at most two non-interrupted jobs executed in time interval $(L_h, L_h + e_h]$ in schedule $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$. If $(L_h, L_h + e_h]$ is entirely in the execution of a non-interrupted job in schedule $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$, it is clear that we reach the contradiction; otherwise, let job $\hat{J}_u$ be the first non-interrupted job whose execution time interval intersects $(L_h, L_h + e_h]$. If there is a non-interrupted job $\hat{J}_v$ executed later than $\hat{J}_u$, we can reach contradiction by the same arguments in the previous paragraph; otherwise, we can reach the contradiction by the same arguments in the first case when there is only one non-interrupted job in schedule $S^{\mathrm{EDF}}_{\hat{\mathbb{J}}^{\mathrm{LEF}}}$. □

In the following, we prove that the cardinality of job set $\hat{\mathbb{J}}^*_2$ is at most twice of that of job set $\hat{\mathbb{J}}^{\mathrm{LEF}}$. The proof process is an iterative process. In each iteration, the process conducts a time domain revision which removes one selected time interval from current time domain. In the mean time, we show that only one job in job set $\hat{\mathbb{J}}^{\mathrm{LEF}}$ is scheduled in the removed time interval and at most two jobs in job set in $\hat{\mathbb{J}}^*_2$ are scheduled in the removed time interval. When the process terminates, we show that only one job in job set $\hat{\mathbb{J}}^{\mathrm{LEF}}$ is scheduled in the remaining time domain and at most two jobs in job set $\hat{\mathbb{J}}^*_2$ are scheduled in the remaining time domain.

In i-th iteration, the proof process selects one job, denoted by $\hat{K}^i$, in job set $\hat{\mathbb{J}}^{\text{LEF}}$, updates the set of jobs scheduled by Algorithm LEF, denoted by $\hat{\mathbb{K}}^{\text{LEF},i}$ where $\hat{\mathbb{K}}^{\text{LEF},0} \equiv \hat{\mathbb{J}}^{\text{LEF}}$, and updates the set of jobs scheduled by an arbitrary algorithm, denoted by $\hat{\mathbb{K}}^{*,i} = \hat{\mathbb{K}}_1^{*,i} \cup \hat{\mathbb{K}}_2^{*,i}$ where $\hat{\mathbb{K}}_1^{*,0} \equiv \hat{\mathbb{J}}^* \cap \hat{\mathbb{J}}^{\text{LEF}}$ and $\hat{\mathbb{K}}_2^{*,0} \equiv \hat{\mathbb{J}}^* \setminus \hat{\mathbb{J}}^{\text{LEF}}$. The selected job $\hat{K}^i$ is the job which is in job set $\hat{\mathbb{K}}^{\text{LEF},i-1}$ and whose execution time is the least among all the other *un-interrupted* jobs in job set $\hat{\mathbb{K}}^{\text{LEF},i-1}$. By definitions, we know that job $\hat{K}^i$ executes from start to completion in the time interval $(s_{K_i}, c_{K_i}]$. In the EDF schedule for job set $\hat{\mathbb{K}}_2^*$, the jobs executed at the time instants s_{K_i} and c_{K_i} are denoted by $\hat{K}^{\ell_i}$ and $\hat{K}^{r_i}$, respectively. ($\hat{K}^{\ell_i}$ and $\hat{K}^{r_i}$ may or may not exist.)

We revise the time domain by treating s_{K_i} and c_{K_i} as the same time instant. The start time L_q and end time R_q of the selected feasible interval for any job $\hat{K}_q$ in job sets $\hat{\mathbb{K}}^{\text{LEF},i} \setminus \{\hat{K}^i\}$ and $\hat{\mathbb{K}}_2^{*,i} \setminus \{\hat{K}^{\ell_i}, \hat{K}^{r_i}\}$ are revised according to the following rules:

Rule (1): if $R_q \le s_{K_i}$, L_q and R_q remain unchanged;
Rule (2): if $L_q \le s_{K_i} \le R_q \le c_{K_1}$, L_q remains unchanged and R_q is revised as s_{K_i};
Rule (3): if $L_q \le s_{K_i} \le c_{K_i} \le R_q$, L_q remains unchanged and R_q is revised as R_q minus $(c_{K_i} - s_{K_i})$;
Rule (4): if $s_{K_i} \le L_q \le c_{K_i} \le R_q$, L_q is revised as s_{K_i} and R_q is revised as R_q minus $(c_{K_i} - s_{K_i})$;
Rule (5): if $c_{K_i} \le L_q$, L_q is revised as L_q minus $(c_{K_i} - s_{K_i})$ and R_q is revised as R_q minus $(c_{K_i} - s_{K_i})$.

Note that $s_{K_i} < L_q \le R_q \le c_{K_i}$ and $s_{K_i} \le L_q \le R_q < c_{K_i}$ are not listed above. This is because these two cases hold only if the execution time of job $\hat{K}_q$ is less than that of job $\hat{K}^i$, which contradicts Lemma 2. After revising the time domain, the process updates job sets $\hat{\mathbb{K}}^{\text{LEF},i}$, which is $\hat{\mathbb{K}}^{\text{LEF},i-1} \setminus \hat{K}_i$, and $\hat{\mathbb{K}}_2^{*,i}$, which is $\hat{\mathbb{K}}_2^{*,i-1} \setminus \{\hat{K}^{\ell_i}, \hat{K}^{r_i}\}$. The process terminates when i is $|\hat{\mathbb{K}}^{\text{LEF}}|$.

We show that the following properties hold for any $i = 0, 1, \ldots, |\hat{\mathbb{J}}^{\text{LEF}|-1}$:

(P4). Job sets $\hat{\mathbb{K}}_2^{*,i}$ and $\hat{\mathbb{J}}^{\text{LEF},i}$ are both schedulable.
(P5). For any job $\hat{K}_h$ in job set $\hat{\mathbb{K}}_2^{*,i}$, the job set $\{\hat{K}_j \mid \hat{K}_j \in \hat{\mathbb{K}}^{\text{LEF},i}$ and $e_j \le e_h\} \cup \{\hat{K}_h\}$ is not schedulable. (This property is similar to Lemma 1.)
(P6). The execution time of any job in job set $\hat{\mathbb{K}}_2^{*,i}$ is no less than the execution time of job $\hat{K}^{i+1}$ if $\hat{\mathbb{K}}_2^{*,i}$ is not an empty set. (This property is similar to Lemma 2.)

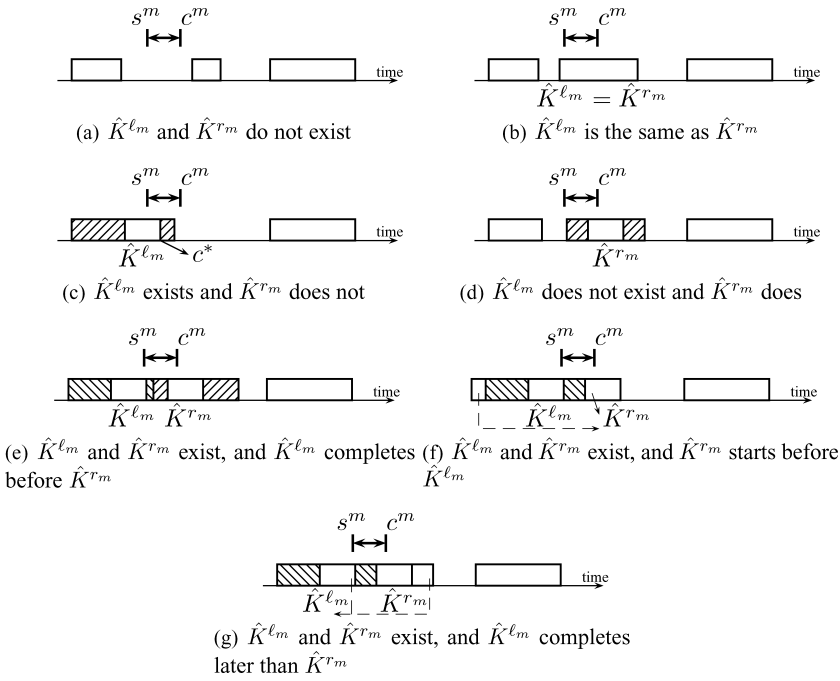
By definitions, we know that both job sets $\hat{\mathbb{K}}_2^{*,0}$ and $\hat{\mathbb{K}}^{\text{LEF},0}$ are schedulable. Therefore, with Lemmas 1 and 2, we know that the properties (P4), (P5), and (P6) hold when $i = 0$. We prove these properties by induction.

Lemma 3. *Suppose that properties (P4), (P5), and (P6) hold when $i = 0, \ldots, m-1$. Then, properties (P4), (P5), and (P6) also hold when $i = m$.*

Proof: Properties (P5) and (P6) are proved by the same arguments in the proofs of Lemmas 1 and 2 when job sets $\hat{\mathbb{K}}_2^{*,m-1}$ and $\hat{\mathbb{K}}^{\text{LEF},m-1}$ are schedulable, and the properties (P5) and (P6) hold for $i = 0, \ldots, m-1$.

We focus on proving Property (P4). The schedulability is trivial for job set $\hat{\mathbb{K}}^{\text{LEF},m}$, because we revise the time domain according to the non-interrupted job $\hat{K}^m$ in job set $\hat{\mathbb{K}}^{\text{LEF},m}$. Hence, it is sufficient to only show the schedulability of job set $\hat{\mathbb{K}}_2^{*,m}$.

Let s^m and c^m be the start time and completion time of job $\hat{K}^m$ in the EDF schedule of job set $\hat{\mathbb{K}}^{\text{LEF},m-1}$, respectively. By definitions, the EDF schedule of job set $\hat{\mathbb{K}}^{\text{LEF},m-1}$ completes



(a) $\hat{K}^{\ell_m}$ and $\hat{K}^{r_m}$ do not exist

(b) $\hat{K}^{\ell_m}$ is the same as $\hat{K}^{r_m}$

(c) $\hat{K}^{\ell_m}$ exists and $\hat{K}^{r_m}$ does not

(d) $\hat{K}^{\ell_m}$ does not exist and $\hat{K}^{r_m}$ does

(e) $\hat{K}^{\ell_m}$ and $\hat{K}^{r_m}$ exist, and $\hat{K}^{\ell_m}$ completes before $\hat{K}^{r_m}$

(f) $\hat{K}^{\ell_m}$ and $\hat{K}^{r_m}$ exist, and $\hat{K}^{r_m}$ starts before $\hat{K}^{\ell_m}$

(g) $\hat{K}^{\ell_m}$ and $\hat{K}^{r_m}$ exist, and $\hat{K}^{\ell_m}$ completes later than $\hat{K}^{r_m}$

Fig. 5 Illustrative examples for seven different schedules in schedule $S^{\text{EDF}}_{\hat{\mathbb{K}}_2^{*,m-1}}$

all the jobs in job set $\hat{\mathbb{K}}^{\text{LEF},m-1}$ in time. There are seven different possible schedules for time interval $(s^m, c^m]$ in schedule $S^{\text{EDF}}_{\hat{\mathbb{K}}_2^{*,m-1}}$. In the following, we discuss the schedulability for each case.

Case (a): Figure 5(a) illustrates the case that no job is scheduled at time instants s^m and c^m. In this case, we know that no job is executed in time interval $(s^m, c^m]$ in the EDF schedule for job set $\hat{\mathbb{K}}_2^{*,m-1}$ because of the property (P6) for $i = m - 1$. In other words, job set $\hat{\mathbb{K}}_2^{*,m-1}$ and $\hat{\mathbb{K}}_2^{*,m}$ are equal. Hence, job set $\hat{\mathbb{K}}_2^{*,m}$ is schedulable in this case.

Case (b): Figure 5(b) illustrates the case that one job is scheduled at both time instant s^m and c^m. In other words, $\hat{K}^{\ell_m}$ and $\hat{K}^{r_m}$ refer to the same job. Because no job would preempt job $\hat{K}^{\ell_m}$ in time interval $(s^m, c^m]$ according to the property (P6) when $i = m - 1$, removing job $\hat{K}^{\ell_m}$ does not postpone the schedule for any other job. Job set $\hat{\mathbb{K}}_2^{*,m}$ is still schedulable.

Case (c): Figure 5(c) illustrates the case that job $\hat{K}^{\ell_m}$ executes at time instant s_m but no job executes at time instant c_m. (In other words, job $\hat{K}^{r_m}$ does not exist.)

We know that all of the jobs executed in time interval $(s^m, c^m]$ in schedule $S^{\text{EDF}}_{\hat{\mathbb{K}}_2^{*,m-1}}$, except for job $\hat{K}^{\ell_m}$, are preempted by job $\hat{K}^{\ell_m}$ according to the EDF property (P3) and start their executions before the start time of job $\hat{K}^{\ell_m}$. Suppose that job $\hat{K}^{\ell_m}$ completes at the time instant c^*, which is no greater than c^m. Hence, job $\hat{K}^{\ell_m}$ executes for $e^{\ell_m} - (c^* - s^m)$ units of time before time s^m, where e^{ℓ_m} is the execution time for job $\hat{K}^{\ell_m}$. Let the amount of non-idle time in time interval $(c^*, c^m]$ in schedule $S^{\text{EDF}}_{\hat{\mathbb{K}}_2^{*,m-1}}$ be x. We know that $x + (c^* - s^m) \leq c^m - s^m \leq e^{\ell_m} = (c^* - s^m) + e^{\ell_m} - (c^* - s^m)$, where the second inequality comes from Property (P6). Therefore, $x \leq e^{\ell_m} - (c^* - s^m)$.

By swapping the schedule in time interval $(c^*, c^m]$ and that in time interval executing job $\hat{K}^{\ell_m}$ before s^m in schedule $S^{\text{EDF}}_{\hat{\mathbb{K}}_2^{*,m-1}}$, we can derive a schedule to complete all of the jobs in job set $\hat{\mathbb{K}}_2^{*,m-1} \setminus \{\hat{K}^{\ell_m}\}$ in time before time s^m. This is because schedule $S^{\text{EDF}}_{\hat{\mathbb{K}}_2^{*,m-1}}$ is feasible and no job starts its execution earlier than its original start time or completes later than its original completion time. Therefore, job set $\hat{\mathbb{K}}_2^{*,m} = \hat{\mathbb{K}}_2^{*,m-1} \setminus \{\hat{K}^{\ell_m}\}$ is schedulable for Case (c).

Case (d): Figure 5(d) illustrates the case that no job is scheduled at time s^m and job $\hat{K}^{r_m}$ executes at time c^m. (In other words, job $\hat{K}^{\ell_m}$ does not exist.) Case (d) can be proved by the same argument for Case (c).

Case (e): Figure 5(e) illustrates the case that jobs $\hat{K}^{\ell_m}$ and $\hat{K}^{r_m}$ exist and are not equal, and job $\hat{K}^{\ell_m}$ completes before job $\hat{K}^{r_m}$ starts in schedule $S^{\text{EDF}}_{\hat{\mathbb{K}}_2^{*,m-1}}$. The schedulability can be proved by combining the analysis for Case (c) and (d).

Case (f): Figure 5(f) illustrates the case that jobs $\hat{K}^{\ell_m}$ and $\hat{K}^{r_m}$ exist and are not equal, and job $\hat{K}^{r_m}$ starts before job $\hat{K}^{\ell_m}$ starts in schedule $S^{\text{EDF}}_{\hat{\mathbb{K}}_2^{*,m-1}}$. (In other words, job $\hat{K}^{\ell_m}$ preempts job $\hat{K}^{r_m}$.)

For this case, the completion time c^* for job $\hat{K}^{\ell_m}$ is earlier than that of job $\hat{K}^{r_m}$, where $c^* < c^m$. It is not difficult to show that all the jobs executed in time interval $(c^*, c^m]$ in schedule $S^{\text{EDF}}_{\hat{\mathbb{K}}_2^{*,m-1}}$ are preempted by job $\hat{K}^{\ell_m}$. Advancing the job executions in time interval $(c^*, c^m]$, except for that for job $\hat{K}^{r_m}$, to the time interval for executing job $\hat{K}^{\ell_m}$ before time s^m and removing the job executions for jobs $\hat{K}^{\ell_m}$ and $\hat{K}^{r_m}$ would derive a schedule to complete all of the jobs in job set $\hat{\mathbb{K}}_2^{*,m-1} \setminus \{\hat{K}^{\ell_m}, \hat{K}^{r_m}\}$ in time without executing any job in time interval $(s^m, c^m]$.

Case (g): Figure 5(g) illustrates the case that jobs $\hat{K}^{\ell_m}$ and $\hat{K}^{r_m}$ exist and are not equal, and job $\hat{K}^{\ell_m}$ completes after job $\hat{K}^{r_m}$ completes in schedule $S^{\text{EDF}}_{\hat{\mathbb{K}}_2^{*,m-1}}$. (In other words, job $\hat{K}^{r_m}$ preempts job $\hat{K}^{\ell_m}$.) Similar arguments for Case (f) could also be made for the case.

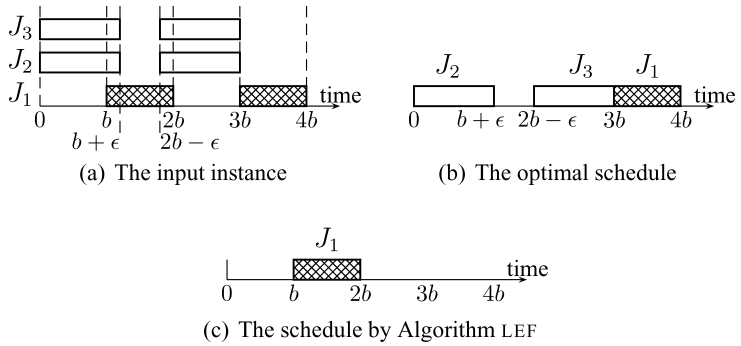
Hence, properties (P4), (P5), and (P6) hold when $i = m$. □

Now, we show the approximation factor for Algorithm LEF.

Theorem 3. *Algorithm* LEF *is a 3-approximation algorithm for the IJS problem.*

Proof: It suffices to prove $|\mathbb{J}^{\text{LEF}}| \geq 2|\mathbb{J}_2^*|$ by showing that there are at most two jobs in job set $\hat{\mathbb{K}}_2^{*,|\mathbb{J}^{\text{LEF}}|-1}$ after we have removed at most two jobs to construct job set $\hat{\mathbb{K}}_2^{*,i}$ from job set $\hat{\mathbb{K}}_2^{*,i-1}$ for $i = 1, 2, \ldots, |\mathbb{J}^{\text{LEF}|-1}$. By properties (P5) and (P6), we know that the EDF schedule of job set $\hat{\mathbb{K}}_2^{*,|\mathbb{J}^{\text{LEF}}|-1}$ could only schedule at most two jobs: one starts before the start time of the selected feasible interval of job $\hat{K}^{|\mathbb{J}^{\text{LEF}}|}$ in $\hat{\mathbb{K}}^{*,|\mathbb{J}^{\text{LEF}}|-1}$, and another one starts before the start time of the feasible interval of job $\hat{K}^{|\mathbb{J}^{\text{LEF}}|}$ plus the execution time of $\hat{K}^{|\mathbb{J}^{\text{LEF}}|}$. With Lemmas 1, 2, and 3, we know that Algorithm LEF is a 3-approximation algorithm for the IJS problem. □

The tightness of Algorithm LEF is demonstrated by the following example: Consider $\mathbb{J} = \{J_1, J_2, J_3\}$, where $J_1 = (b, \{(b, 2b], (3b, 4b]\})$, $J_2 = (b + \epsilon, \{(0, b + \epsilon], (2b - \epsilon, 3b]\})$, and $J_3 = (b + \epsilon, \{(0, b + \epsilon], (2b - \epsilon, 3b]\})$ ($b > 0$ and $0.5b \geq \epsilon > 0$), as illustrated in Fig. 6(a). The optimal schedule, referred to Fig. 6(b), is to execute J_1, J_2, and J_3 in their second, first, and second feasible intervals, respectively, whereas Algorithm LEF only schedules job J_1 in its first feasible interval, shown as Fig. 6(c).



(a) The input instance (b) The optimal schedule

(c) The schedule by Algorithm LEF

Fig. 6 A tight example for Algorithm LEF

5. Performance evaluation

The experiments described in this section are meant to evaluate the performance of the proposed algorithms. For non-preemptible jobs, the performance of our proposed Algorithm LECF is compared with Algorithm First-Come-First (FCF), which selects the jobs in a non-decreasing order of their start times of the first feasible interval and then schedules jobs as early as possible after the completion time of those scheduled jobs. For preemptible jobs, we compare the performance of Algorithm LEF against those algorithms in Shih et al. (2003), including Fewer Feasible Interval First-based (FFIF-First Fit, FFIF-Last Fit, FFIF-Worst Fit, FFIF-Best Fit) and First Come First Serve-based algorithms (FCFS-First Fit, FCFS-Last Fit, FCFS-Worst Fit, and FCFS-Best Fit).

5.1. Workload generation and performance metrics

The job sets in our simulations are generated based on two parameters: the number of jobs and average arrival rate. The former is the number of jobs in the job set; the latter is the average number of jobs arrived within each second in Possion distribution. Each job is characterized by four parameters: execution time, number of feasible intervals, length of each feasible interval, and *temporal distance* between two consecutive feasible intervals. By temporal distance, we mean the difference between the start time of feasible interval $I_{i,j}$ of job J_i and the end time of feasible interval $I_{i,j-1}$ of job J_i for $j > 1$.

Two types of workloads are generated in the evaluations. For Type I workload, we simulate the job sets which have small number of jobs and in which each job has small number of feasible intervals. For Type II workload, we simulate the job sets which have large number of jobs and each job has large number of feasible intervals. The workload parameters for two types of workload are listed in Table 1. For Type I workload, performing exhaustive search on the generated job sets for optimal solutions could be done efficiently, i.e., within several minutes for an input instance. Experimental results are conducted from 512 independent experiments for each parameter configuration.

We use two metrics to measure the performance of the algorithms: completion rate and normalized completion rate. *Completion rate* for an input instance is defined as the ratio of the number of jobs completed in time in the derived schedule to that in the optimal solution. When the optimal solution for an input instance could be derived efficiently, the measurements on completion rates of the evaluated algorithms provide performance indexes.

Table 1 Simulation parameters for type I and type II

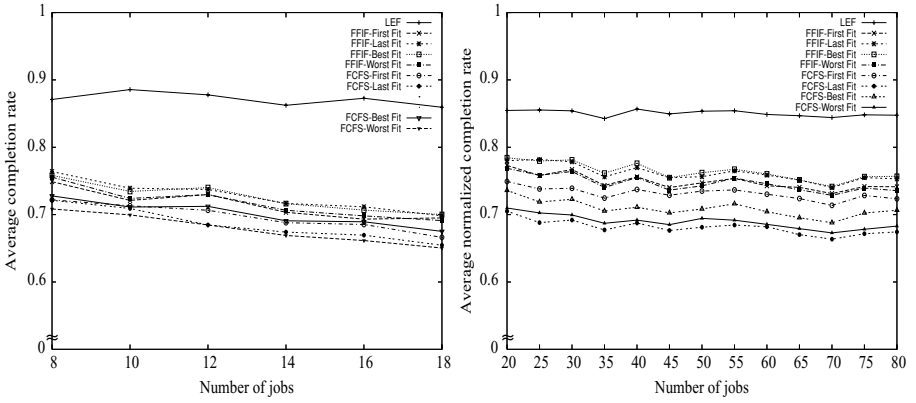
Parameters	Type I Workload	Type II Workload
Number of jobs	8, 10, 12, 14, 16, 18	20 to 80 stepped by 5
Average arrival rate (Poisson)	4	2
Execution time	uniform(200, 400) ms	uniform(100, 500) ms
Number of feasible intervals	uniform(1, 3)	uniform(1, 5)
Interval length*	uniform(max{200, e}, 500) ms	uniform(max{200, e}, 600) ms
Temporal distance	uniform(100, 300) ms	uniform(100, 300) ms

* e denotes the execution time of the job determined.

Normalized completion rate for an input instance is the ratio of the number of jobs completed in time in the derived schedule to the number of jobs in the input job set. When the optimal solution for an input instance could not be derived efficiently, normalized completion rate is used to compare the performance for different algorithms.

5.2. Experimental results

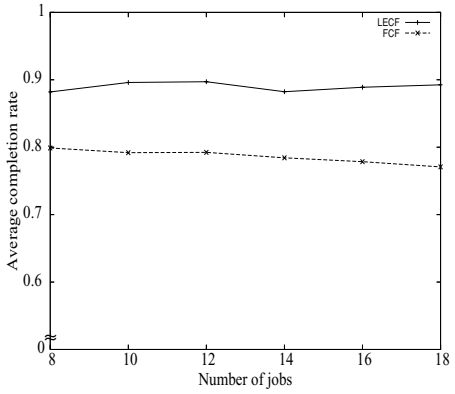
For preemptible jobs, Figs. 7(a) and 7(b) show the evaluation results for Type I and Type II workloads. In Fig. 7(a), the average completion rates of Algorithm LEF range from 86% to 89%, while the average completion rates of the FFIF-based algorithms and the FCFS-based algorithms range from 69% to 77% and 65% to 73%, respectively. Algorithm LEF outperforms the FFIF-based algorithms and the FCFS-based algorithms. It is because schedules generated from Algorithm LEF are based on a highly related factor, i.e., the execution time, with optimized solutions, whereas the schedules generated from FFIF-based and FCFS-based algorithms are based on the number of feasible intervals and the arrival time of jobs which are not highly related factors. For Type I workloads, the last fit strategy and best fit strategy for FFIF-based algorithms perform better than the other strategies do, while the first fit strategy and best fit strategy for FCFS-based algorithms perform better than the other



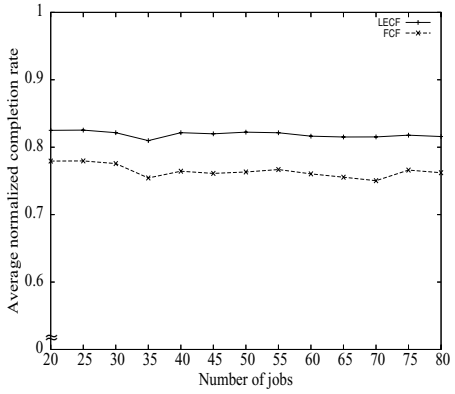
(a) Average completion rate for Type I workload

(b) Average normalized completion rate for Type II workload

Fig. 7 Evaluation results for preemptible jobs.



(a) Average completion rate for Type I workload



(b) Average normalized completion rate for Type II workload

Fig. 8 Evaluation results for non-preemptible jobs

strategies do. Furthermore, the performance of Algorithm LEF is more steady in Fig. 7(a), in which the average completion rates of FFIF-based algorithms and FCFS-based algorithms decrease when the number of jobs increases. In Fig. 7(b), the average normalized completion rates of Algorithm LEF, the FFIF-based algorithms, and the FCFS-based algorithms range from 84% to 86%, 73% to 78%, and 66% to 75%, respectively. The trends of the simulation results in Fig. 7(a) are similar to those in Fig. 7(b).

For non-preemptible jobs, the simulation results are presented in Figs. 8(a) and 8(b) for Type I and Type II workloads by measuring the average completion rate and the average normalized completion rate, respectively. The average completion rates for Algorithm LECF and Algorithm FCF range from 87% to 90% and 77% to 80%, respectively. The average normalized completion rates for Algorithm LECF and Algorithm FCF range from 81% to 83% and 75% to 78%, respectively. The results indicate that Algorithms LECF and LEF not only guarantee the approximation factors but also outperform other multiple feasible interval scheduling algorithms.

6. Conclusion

In this paper, we present two approximation algorithms for the multiple feasible interval jobs. For preemptible jobs, Algorithm LECF is shown being a 2-approximation algorithm by executing the job which is able to complete in time earliest among the unselected jobs in a greedy manner. By adopting the heuristic to choose the preemptible job with the least execution time, Algorithm LEF is proved being a 3-approximation algorithm for scheduling preemptible jobs. Both of the two proposed algorithms are also shown with tight approximation factors by presenting tight input instances. The proposed algorithms are evaluated by extensive simulations with performance comparisons against the proposed algorithms in previous study. The results show that Algorithm LECF and Algorithm LEF could complete more jobs in time than the previous heuristic algorithms could.

For future research, we would further explore algorithms with less runtime overhead. We would also explore the scheduling problem to maximize the sum of the rewards of completed jobs in the multiple feasible interval job model when every job is associated with a reward value for its completion.

Acknowledgments We would like to thank Prof. Tei-Wei Kuo at National Taiwan University for his valuable inputs and the reviewers for their valuable feedbacks. This work is supported in part by a grant from the NSC program 93-2752-E-002-008-PAE and in part by a grant from the NSC program 93-2213-E-002-090.

References

Aydin H, Melhem R, Mosse D, Alvarez P (1999) Optimal reward-based scheduling for periodic real-time tasks. In: Proceedings of the 20th IEEE Real-Time Systems Symposium pp 79–89

Sprunt B, Sha L, Lehoczky J (1989) Aperiodic task scheduling for hard-real-time systems. Real-time Systems Journal pp 27–60

Baptiste P (1999) An $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. Operations Research Letters 24:175–180

Han C-C, Lin K-J (1992) Scheduling distance-constrained real-time tasks. In: Proceedings of the 13th IEEE Real-Time Systems Symposium pp 300–308

Carlier J (1982) The one machine sequencing problem. European Journal of Operational Research 11:42–47

Cheong IK (1992) Scheduling Imprecise Hard Real-Time Jobs with Cumulative Error. PhD thesis, University of Illinois at Urbana-Champaign

Hamdaoui M Ramanathan P (1995) A dynamic priority assignment technique for streams with (m, k)-RM deadlines. IEEE Transaction on Computers 44(12):1443–1451

Koren G, Shasha D (1995) Skip-over: Algorithms and complexity for overloaded systems that allow skips. In: Proceedings of the 16th IEEE Real-Time Systems Symposium pp 110–117

Lawler EL (1990) A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. Operations Research 26(1):125–133

Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM 20(1):46–61

Liu JW-S (2000) Real-time systems. Prentice Hall, Englewood, Cliffs, NJ

Liu JW-S, Lin K-J, Shih WK, Yu AC-S, Chung J-Y, Zhao W (1991) Algorithms for scheduling imprecise computations. IEEE Computer 24(5):58–68

Quan G, Hu, X (2000) Enhanced fixed-priority scheduling with (m, k)- firm guarantee. In: Proceedings of the 21th IEEE Real-Time Systems Symposium pp 79–88

Shih C-S, Liu JW-S, Cheong IK (2003) Scheduling jobs with multiple feasible intervals. In: Proceedings of the Real-Time and Embedded Computing Systems and Applications pp 53–71

Shih WK, Liu JW-S (1995) Algorithms for scheduling imprecise computations with timing constraints to minimize maximum error. IEEE Transaction on Computers 44(3):466–471

Shih WK, Liu JW-S, Chung JY (1991) Algorithms for scheduling imprecise computations with timing constraints. SIAM Journal on Computing 20(3):537–552

Vazirani VV (2001) Approximation algorithms. Springer