Approximation Algorithms for Scheduling Multiple Feasible Interval Jobs

Jian-Jia Chen  
Department of Computer Science and Information Engineering  
National Taiwan University, Taiwan, ROC  
Email: r90079@csie.ntu.edu.tw

Jun Wu  
Department of Management of Information Systems  
Chung Kuo Institute of Technology, ROC  
Email: junwu@mail.ckitc.edu.tw

Chi-Sheng Shih and Tei-Wei Kuo  
Department of Computer Science and Information Engineering  
Graduate Institute of Networking and Multimedia  
National Taiwan University, Taiwan, ROC  
Email: {cshih, ktw}@csie.ntu.edu.tw

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. 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 \( \text{LECF} \) is with a 2-approximation factor; when jobs are preemptible, Algorithm \( \text{LEF} \) is proved being a 3-approximation algorithm. We also show that our analysis on the two algorithms is tight by providing a set of input instances. Simulation results demonstrate that Algorithms \( \text{LECF} \) and \( \text{LEF} \) not only guarantee the approximation factors but also outperform other multiple feasible interval scheduling algorithms.

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 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 in effect 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 is predictable. Hence, the starts and ends of all feasible intervals can be determined sufficiently accurately 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 the point in time 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. Moreover, the optional jobs in the error-cumulative imprecise computation model studied in \([7], [9]\) are also examples of jobs with multiple feasible intervals \([8]\).

This work is supported in part by grants from the NSC program 93-2752-E-002-008-PAE and the NSC program 93-2213-E-002-090.

Shih, Liu, and Cheong \([8]\) 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 an optimal algorithm with exponential-time complexity and a family of heuristics in polynomial time for the \( \mathcal{NP} \)-complete problem. 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 \([5]\), reward-based model \([1]\), (error-cumulative) imprecise computation model \([7], [9]\), and (mu)-firm guarantee model \([4]\). 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 \([2], [6]\).

In this paper, we develop two algorithms providing worst-case guarantees. Algorithm \( \text{LECF} \) schedules non-preemptible multiple feasible interval jobs. Theoretical analysis shows that Algorithm \( \text{LECF} \) is with a 2-approximation factor, in which the number of multiple feasible interval jobs completed in time in the derived schedule from Algorithm \( \text{LECF} \) is at least one half of the number of any non-preemptive schedule. On the other hand, Algorithm \( \text{LEF} \) is designed for the case that jobs are preemptible. The approximation factor for Algorithm \( \text{LEF} \) is shown being 3. We also show the tightness of our analysis on the performance guarantees by providing a set of input instances. Simulation results show that Algorithms \( \text{LECF} \) and \( \text{LEF} \) 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. Section 3 presents a 2-approximation algorithm for non-preemptible jobs, while Section 4 gives a 3-approximation algorithm for preemptible jobs. Section 5 shows simulation results of our algorithms against existing approaches. Section 6 is the conclusion.

2. 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. 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 \( \alpha \), is the amount of time required to complete the job when it executes alone and has all the resources it requires. Associated with each job is a set of disjointed time intervals, called feasible intervals. The job can execute only in 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.

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 feasible 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 \( I_i = \{I_{i,1}, I_{i,2}, \ldots, I_{i,n(i)}\} \), 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 \( J_i = (e_i, l_i) \). 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 for jobs with multiple feasible intervals. We call a failed attempt to complete the execution of a job in one of its feasible intervals a deadline miss. A deadline miss occurs when a job executing in a feasible interval remains incomplete at the end of the feasible interval. When a job misses the deadline for its last feasible interval, the job fails to meet its timing constraint. We state the timing constraint of a job as follows.

**Definition 1 (In-Time Completion):** An execution of a job \( J \) completes in time if and only if one of its execution completes in time.

A schedule for a job set \( J \) denoted by \( S_j \), is the time sequence of executing jobs in the job set. For a given schedule, the starting 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 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 un-preempted in the schedule; otherwise, the job is preempted 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.

This paper is interested in 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. For the **Non-preemptible Interval Job Scheduling (n-IJS) problem**, we are given a job set \( J = \{J_1, J_2, \ldots, J_m\} \) of non-preemptible multiple feasible interval jobs. The objective is to find a feasible schedule for a subset of job set \( J \) such that the size of the subset is maximized. The **Preemptible Interval Job Scheduling (IJS) problem** considers preemptible multiple interval jobs with the same objective of the n-IJS problem. It is not hard to see that the IJS and n-IJS problems are both \( NP \)-hard since finding a schedule to complete all the jobs in a set of interval jobs in time is \( NP \)-complete [8]. We focus on developing polynomial-time approximation algorithms [10, §1]. An algorithm for a maximization problem is with an \( \alpha \)-approximation factor (ratio) [10, §1] for the IJS or n-IJS problem if it guarantees that the number of jobs completed in time in its derived schedule is no less than \( \frac{1}{\alpha} \) times that of an optimal schedule. In this paper, an algorithm with an \( \alpha \)-approximation factor is also referred to as an \( \alpha \)-approximation algorithm.

3. **Scheduling of Non-preemptible Jobs**

Before we present the algorithm for the n-IJS problem, we define several terms used in this section. **Scheduling time** is the time instant at which the scheduler selects one unscheduled 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 \( I_{e}(t) \), for job \( J_i \) is the earliest feasible interval in which job \( J_i \) completes in time, if and only if one of its executions completes in time.

**Algorithm 1 : Least Earliest Completion Time First (LECF)**

**Input:** \( J = \{J_1, J_2, \ldots, J_n\} \)

**Output:** A feasible schedule for a subset \( J_{LECF} \) of \( J \)

1. remove feasible intervals \( I_{e}(t) \) from \( J_i \) for every job \( J_i \in J \) if \( e_i > R_{i,t} \).
2. let scheduling time \( t = 0 \).
3. \( J_{LECF} = \emptyset \).
4. set \( I_{e}(t) \) as the earliest completion interval at scheduling time \( t \) for every job \( J_i \in J \).
5. repeat
6. remove job \( J_{e} \) from \( J \) in which the earliest completion time at scheduling time \( t \) of \( J_{e} \) is the least among all the jobs in \( J \).
7. \( J_{LECF} \leftarrow J_{LECF} \cup \{J_{e}\} \).
8. schedule \( J_{e} \) to start at \( \max(t, L_{e}, e_i) \) and complete at \( \max(t, L_{e}, e_i) + e_i \).
9. update schedule time \( t \) to \( \max(t, L_{e}, e_i) + e_i \).
10. update the earliest completion interval at \( t \) for every job in \( J \) and remove those jobs without earliest completion interval at \( t \) from \( J \).
11. until \( J \) is empty.
12. return the schedule of \( J_{LECF} \).

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 0. Algorithm LECF repeats the following procedures to select a job and advance scheduling time \( t \) until all the unscheduled jobs are unschedulable at scheduling time \( t \). At scheduling time \( t \), we greedily select job \( J_{e} \) for execution in which the earliest completion time at scheduling time \( t \) of \( J_{e} \) is the least among all the unscheduled jobs. \( J_{e} \) executes in the time interval from the maximum value of \( t \) and the start time of interval \( I_{e}, k \) at time \( t \) of \( J_i \), and the execution time of \( J_{e} \), i.e., \( \max(t, L_{e}, e_i) \), where \( L_{e}, k \) is the start time of \( I_{e}, k \). A job \( J \) is said to be unschedulable at scheduling time \( t \) if no earliest completion interval at scheduling time \( t \) exists.

When presenting the algorithm for the n-IJS problem, we define several terms used in this section. **Scheduling time** is the time instant at which the scheduler selects one unscheduled 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 \( I_{e}(t) \), for job \( J_i \) is the earliest feasible interval in which job \( J_i \) completes in time and its starting time is no earlier than \( t \). That is, \( I_{e}(t) \) is the earliest feasible interval of job \( J_i \) satisfying \( R_{i,k} - e_i \geq t \) and \( R_{i,k} - L_{i,k} \geq e_i \). The **earliest completion time** of job \( J_i \) at scheduling time \( t \) is the earliest time instant at which \( J_i \) can complete in time when it starts its execution no earlier than \( t \). When scheduling time \( t \) is in the earliest completion interval \( I_{e}, k \) of job \( J_i \), \( J_i \) can start its execution at time instant \( t \); otherwise, \( J_i \) can start at the time of feasible interval \( I_{e}, k \), i.e., \( L_{e}, k \). 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 interval \( I_{e}, k \) at time \( t \) of \( J_i \) and the execution time of \( J_i \), i.e., \( \max(t, L_{e}, e_i) \), where \( L_{e}, k \) is the start time of \( I_{e}, k \). A job \( J_i \) is said to be unschedulable at scheduling time \( t \) if no earliest completion interval at scheduling time \( t \) exists.

Theorem 1: Algorithm LECF is a 2-approximation algorithm for the n-IJS problem.

**Proof:** Let \( J \) be a subset of \( J \), including an optimal subset, such that there is a feasible schedule \( S' \) for job set \( J \), \( J' \) and \( J'' \) are the intersection of \( J_{LECF} \) and \( J \), i.e., \( J' \equiv J_{LECF} \cap J \), and be
the relative complement of $J_i$ in $J_i^1$, i.e., $J_i^2 \equiv J_i \setminus J_i^1$, respectively. In other words, $J_i^1$ consists of the jobs both in $J_i$ and $J_i^{LECF}$, while $J_i^2$ consists of the jobs that are in $J_i$ but not in $J_i^{LECF}$. Naturally, $|J_i^1| \leq |J_i^{LECF}|$. If $|J_i^2| \leq |J_i^{LECF}|$ is true, we will have $|J_i^{LECF}| \geq 0.5(|J_i^1| + |J_i^2|) = 0.5|J_i^1|$. Hence, it remains to show $|J_i^2| \leq |J_i^{LECF}|$.

Let $t_j$ be the completion time of the $j$-th selected job in the derived schedule of Algorithm LECF. For the brevity, let $t_0 = 0$. That is, in the $j$-th iteration of Algorithm LECF, scheduling time $t$ is $t_{j−1}$ before $t$ is updated. For the derived schedule of $J_i^{LECF}$, there is only one job executed in the each of the time intervals $(t_{j−1}, t_j)$ for $j = 1$ to $|J_i^{LECF}|$. Let $S_2$ be the schedule by removing the executions of the jobs in $J_i^1$ from schedule $S$. $S_2$ is a feasible schedule of $J_i^2$. Note that the jobs in $J_i^2$ in $S_2$ are executed one by one because all the jobs are non-preemptible.

We claim that there is at most one job whose starting time in $S_2$ is in $[t_{j−1}, t_j)$ for $j = 1$ to $|J_i^{LECF}|$. We prove this statement by contradiction. Assume that there exists an interval $[t_{j−1}, t_j)$ for some $1 \leq j \leq |J_i^{LECF}|$ such that at least two jobs start their executions in this time interval in schedule $S_2$. Let $J_i$ be the job whose starting time $s_i$ is the earliest in this interval in schedule $S_2$. Since there is another job started before time $t_j$ in schedule $S_2$, we know that the completion time $s_i + e_i$ of $J_i$ in $S_2$ must be less than $t_j$. Therefore, we have $s_i + e_i < t_j$ and $s_i \geq t_{j−1}$. That is, the earliest completion time of job $J_i$ at scheduling time $t_{j−1}$ is less than that of the job scheduled in $(t_{j−1}, t_j)$ in the schedule for $J_i^{LECF}$. At scheduling time $t_{j−1}$, job $J_i$ is not scheduled at scheduling time $t_{j−1}$ and has an earlier completion time. In the $j$-th iteration, Algorithm LECF should select job $J_i$ to execute instead of the job selected in $J_i^{LECF}$. This contradicts the least earliest completion first schedule. Hence, there is at most one job whose starting time in $S_2$ is in $[t_{j−1}, t_j)$ for $j = 1$ to $|J_i^{LECF}|$.

If there is a job in $J_i^2$ which starts its execution in $(t_i^{LECF}, \infty)$ in $S_2$, this contradicts the termination condition of Algorithm LECF: all the jobs in $J_i \setminus J_i^{LECF}$ are unschedulable at scheduling time $t_i^{LECF}$. As a result, there are at most $|J_i^{LECF}|$ jobs starting their executions in $S_2$. Therefore, we know $|J_i^2| \leq |J_i^{LECF}|$, and Algorithm LECF has a 2-approximation factor.

We now show that our analysis in Theorem 1 is tight. Consider the input instance $J = (J_1, J_2)$, where $J_1 = (b, \{0, b, b + e, 2b + e\})$ and $J_2 = (b - e, \{0, b - e\})$. The optimal schedule is to execute $J_1$ in its second feasible interval and $J_2$ in its first feasible interval. The derived schedule of Algorithm LECF executes only $J_1$ in its first feasible interval.

4. SCHEDULING OF PREEMPTIBLE JOBS

In this section, we deal with the preemptible interval job scheduling (IJS) problem. Algorithm LECF could also be adopted for the IJS problem since a non-preemptive schedule is also feasible for the IJS problem. However, it is not difficult to show that the approximation factor of Algorithm LECF could be $O(|J|)$ for the IJS problem. Therefore, we develop another algorithm with a constant approximation factor based on the earliest-deadline-first (EDF) strategy [2].

A solution to the IJS problem has two parts: feasible interval selection and job scheduling. When feasible interval $I_{j, i}$ has been selected for job $J_i$, we could construct one corresponding job $J_i$ of $J_i$, in which the execution time of $J_i$ is equal to that of $J_i$, and the set of feasible intervals only consists of one feasible interval. Naturally, job $J_i$ can be treated as a traditional real-time jobs: the arrival time and deadline for job $J_i$ are the start and end time of interval $I_{j, i}$, respectively. Suppose that $J_i^1$ is the set of the jobs selected to be executed and $J_i^0$ is the set consisting of the jobs with one feasible interval constructed from the jobs in $J_i^1$. The schedulability of job set $J_i^0$ can be verified by applying the traditional earliest-deadline-first (EDF) strategy: whenever no job is executed, and one job in set $J_i^0$ is available and not completed yet, schedule the available job with the earliest deadline. The optimality of the EDF strategy guarantees that there exists a feasible schedule for job set $J_i^1$ if and only if the schedule derived by the EDF strategy is feasible [3]. We say a set of jobs with selected feasible intervals is schedulable if its EDF schedule is feasible. For the rest of this paper, we denotes the EDF schedule for a set $X$ of jobs as $S_X^{EDF}$ in which every job in $X$ has only one feasible interval.

Therefore, the hardness of the IJS problem comes from the selection of jobs in job set $J_i^1$ and the determination of feasible intervals for the selected jobs. The Least Execution Time First (LEF) algorithm processes the input job sets $J_i$ in a non-descending order according to the execution time of each job. Initially, $J_i^1$ and $J_i^2$ are both empty sets. The job set $J_i$ is initially set to $J_i$ as a working temporal. Let $J_i$ be the job which is associated with the least amount of execution time for all the jobs in job set $T$. If there exists a feasible interval $I_{j, i}$ of job $J_i$, such that it is schedulable for job set $J_i^1$ along with the job $J_i^0$, constructed from $J_i^0$ by selecting $I_{j, i}$ for executions, then job $J_i$ is inserted into job set $J_i$ and job $J_i^0$ is inserted into job set $J_i^0$; otherwise, job $J_i$ is not selected. (We break ties arbitrarily.) By removing job $J_i$ from job set $T$, we repeat the above greedy procedures until job set $T$ becomes an empty set. The pseudo codes of the above algorithm are shown in Algorithm 2, denoted as Algorithm LEF. The schedule derived from Algorithm LEF completes all the jobs in $J_i^1$ in time. Because the EDF schedulability test could be done in $O(|J|)$ if the jobs are sorted by their deadlines in a priori, is it not difficult to see that the time complexity of this algorithm is $O(|J| \sum_{i \in J} |I_i|)$.

\begin{verbatim}
Algorithm 2: Least Execution Time First (LEF)
Input: $J$;
Output: A feasible schedule for a subset $J_i^1$ of $J$;
1: $T \leftarrow J$; $J_i^1 \leftarrow \emptyset$; $J_i^0 \leftarrow \emptyset$;
2: while $T \neq \emptyset$ do
3: let $J_i \leftarrow \arg \min_{J_{i' \in T}} \{e_i\}$;
4: for each $I_{j, i} \in J_i$ do
5: construct $J_i$ by assigning $J_i$ executed in $I_{j, i}$;
6: if $\{J_i\}$ is schedulable by EDF then
7: $J_i^1 \leftarrow J_i^1 \cup \{J_i\}$; $J_i^0 \leftarrow J_i^0 \cup \{J_i\}$;
8: break;
9: $T \leftarrow T \setminus \{J_i\}$;
10: return $S_X^{EDF}$;
\end{verbatim}

In the following discussions, we will show the optimality of Algorithm LEF, i.e., that on the approximation factor. We need the following properties of EDF schedules to prove the approximation factor of Algorithm LEF:

(P1) A set $\hat{J}$ of jobs with selected feasible intervals is schedulable by the EDF strategy if and only if there exists a schedule to complete all the jobs in $\hat{J}$ in time [3].

(P2) If a set $\hat{J}$ of jobs with selected feasible intervals is schedulable by the EDF strategy, then any subset $\hat{J}_i$ of $\hat{J}$ is also schedulable by the EDF strategy, too.

(P3) During the starting time and the completion time of a job in the EDF schedule, there must be at least one un-preempted job.

For the rest of this section, let $J_i^0$ be a subset of $J_i$ in which all the jobs in $J_i^0$ complete in time, while job set $J_i^1$ denotes the set of the jobs with selected feasible intervals by a schedule which can complete all the jobs in $J_i^1$ in time. Let $J_i^0$ be the intersection of $J_i^1$ and $J_i^0$, i.e., $J_i^0 \equiv J_i^1 \cap J_i^0$. and $J_i^2$ be the relative complement of $J_i^0$ in $J_i^1$.
i.e., $J^*_2 \equiv J^* \setminus J_1$. Naturally, $|J^*_1| \leq |J^*_2|$. Let $J^*_2$ be the subset of $J^*$ which consists of the jobs with selected feasible intervals in $J^*_2$. By definitions, the cardinality of job set $J^*_2$ is equal to that of job set $J^*_1$.

The $3$-approximation factor can be shown if we can prove that $|J^*_2|$ is at most twice of $|J^*_1|$. The following lemmas state two important properties for $J^*_1$ and $J^*_2$.

**Lemma 1:** For any job $J_h$ in job set $J^*_2$, the job set consisting of job $J_h$ and all the jobs in job set $J^*_1$ with no greater execution time than that of $J_h$, i.e., $\{J_j | J_j \in J^*_1 \land e_j \leq e_h \} \cup \{J_h\}$, is not schedulable.

**Proof:** This lemma could be proved by contradiction. Suppose that set $J_h \equiv \{J_h\} \cup \{J_j | J_j \in J^*_1 \land e_j \leq e_h\}$ is schedulable for certain job $J_h$ in $J^*_2$. Let $J_h$ be the job with multiple feasible intervals from which we construct job $J_h$. The job set $J^*_2$ is defined to consist of all the jobs in job set $J^*_1$ before we enter the iteration of the while loop to consider job $J_h$ in Algorithm 2. Because job set $J^*_2$ is schedulable and $J^*_2 \cup \{J_h\}$ is a subset of $J_h$, we know that job set $J^*_2 \cup \{J_h\}$ is schedulable based on the EDF properties (P1) and (P2). Algorithm LEF would insert job $J_h$ into job set $J^*_1$. We reach the contradiction.

**Lemma 2:** Suppose that $J^*_1$ is an un-preempted job in the EDF schedule of job set $J^*_1$ with the smallest execution time. If job set $J^*_2$ is not empty, the execution time of any job in $J^*_2$ is no less than that of $J^*_1$.

**Proof:** By the EDF property (P3), we know that such a job $J^*_1$ exists. We prove this lemma by contradiction. Assume that $J^*_2$ is a job in job set $J^*_2$, in which the execution time of $J^*_2$ is less than that of $J^*_1$. We will show that the job $J^*_2$, which constructs $J^*_1$, should be selected in Algorithm LEF. Let $L_m$ denote the start time of the feasible interval of $J^*_2$. By Lemma 1, there must be certain job executed in the time interval $(L_m, L_m + e_h)$ in schedule $S^*_{J^*_1}$. We consider two cases. For the first case, there is at only one un-preempted job in schedule $S^*_{J^*_1}$. For the other case, there are more than one un-preempted job in schedule $S^*_{J^*_1}$.

**First case:** By the EDF property (P3), all of the jobs executed after $L_m$ must be preempted by the un-preempted job $J^*_1$ executed in schedule $S^*_{J^*_1}$. All of the jobs in job set $J^*_1$, excluding job $J^*_1$, start their executions before the starting time $s_j$ of $J^*_1$, and complete their executions after the completion time of $J^*_1$, i.e., $s_j + e_j$. If $s_j \geq L_m$, we could postpone all of the execution pieces of the preempted jobs executed in time interval $(L_m, L_m + e_h)$ in schedule $S^*_{J^*_1}$ to time interval $(s_j, s_j + e_j)$, e.g., referred to Figures 1(a) and 1(b). Similarly, If $s_j < L_m$, we could advance all of the execution pieces of the preempted jobs executed in time interval $(L_m, L_m + e_h)$ in schedule $S^*_{J^*_1}$ to time interval $(s_j, s_j + e_j)$, e.g., referred to Figures 1(c) and 1(d). Since $e_h > e_j$, we know that job set $J^*_2 \setminus \{J^*_1\}$ is schedulable even when no job is executed in time interval $(L_m, L_m + e_h)$. Therefore, we could know that job set $J^*_2 \setminus \{J^*_1\} \cup \{J_h\}$ is schedulable by further scheduling job $J^*_1$ in time interval $(L_m, L_m + e_h)$. By the EDF properties (P1) and (P2), we know that job set $\{J_j | J_j \in J^*_1 \land e_j \leq e_h \} \cup \{J_h\}$ is also schedulable. Algorithm LEF would insert $J_h$ into $J^*_1$. We reach the contradiction.

**Second case:** The proof is very similar to that in the first case when no un-preempted job starts executions before $L_m$ or completes executions after $L_m$ in schedule $S^*_{J^*_1}$. We consider the other cases. Two sub-cases have to be taken: (1) no un-preempted job in schedule $S^*_{J^*_1}$ is executed in time interval $(L_m, L_m + e_h)$; (2) some un-preempted job in schedule $S^*_{J^*_1}$ is executed in time interval $(L_m, L_m + e_h)$. For the first sub-case, let job $J^*_2$ be the first un-preempted job starting after $L_m$ and $J^*_1$ be the last un-preempted job starting before $L_m$, where $s_n$ and $s_n$ denotes the starting times of $J^*_n$ and $J^*_n$ in $S^*_{J^*_1}$, respectively.

**Example:** An illustrative example for the proof in the first case in Lemma 2

We now prove that $|J^*_2|$ is at most twice of $|J^*_1|$. We will show that we can recursively remove one job from job set $J^*_1$ and remove at most two jobs from job set $J^*_2$ so that the resulting job sets are still schedulable even after a transformation of the time domain (we will specify the how later), and the properties stated in Lemmas 1 and 2 could hold for the new job sets.

For non-ambiguity, we introduce the variable $K$ to denote a job with a selected feasible interval after a transformation of the time domain, while $J^*_{K}$ denotes a set of such jobs. Job set $K^*_{1}$ is initialized as a working copy of job set $J^*_1$, while job set $K^*_{2}$ is of job set $J^*_2$. Let $K^*_{1}$ be the job in job set $K^*_{1}$ in which the execution time of $K^*_{1}$ is the smallest among the un-preempted jobs in schedule $S^*_{J^*_1}$. By definitions, job $K^*_{1}$ is executed entirely in the time interval $[s_1, c^1]$, where $c^1 - s_1$ is equal to the execution time $e_1$ of job $K^*_{1}$. Furthermore, let $K^*_{2}$ and $K^*_{2}$ be the jobs in job set $K^*_{2}$ which are executed at the time instants $s_{2}$ and $c^2$ in its EDF schedule, respectively. $(K^*_{1} \text{ and } K^*_{2} \text{ might be empty.})$ Suppose that the time interval $[s_1, c^1]$ can not be used by any jobs in job sets $K^*_{1}$ \{ $K^*_{1}$ \} and $K^*_{2}$ \{ $K^*_{1}$, $K^*_{2}$ \}. We could modify the time domain by treating $s_1$ and $c^1$ as the same time instant. The start time $L_m$ and end time $R_m$ of the feasible interval of a job $K^*_{2}$ in job sets $K^*_{2}$ \{ $K^*_{1}$ \} and $K^*_{2}$ \{ $K^*_{1}$, $K^*_{2}$ \} are revised as follows:

- if $R_m \leq s_1$, then $L_m$ and $R_m$ remain;
- if $L_m \leq s_1 \leq R_m \leq c^1$, then $L_m$ remains, and $R_m$ is revised as $s_1$;
- if $s_1 \leq L_m \leq c^1 \leq R_m$, then $L_m$ remains, and $R_m$ is revised as $R_m$ minus $(c^1 - s_1)$;
- if $s_1 \leq L_m \leq c^1 \leq R_m$, then $L_m$ is revised as $s_1$, and $R_m$ is revised as $R_m$ minus $(c^1 - s_1)$;
- if $c^1 \leq L_m$, then $L_m$ is revised as $L_m$ minus $(c^1 - s_1)$, and $R_m$ is revised as $R_m$ minus $(c^1 - s_1)$.

Note that, if $s_1 \leq L_m \leq R_m \leq c^1$ or $s_1 \leq L_m \leq R_m \leq c^1$, the execution time of job $K_m$ must be less than that of job $K^*_{1}$, which contradicts Lemma 2. The resulting job set of $K^*_{1}$ by excluding job
in the new time domain is referred to job set $K^1$. Let job set $K_2^1$ be the job set of $K_2$ after excluding jobs $K^1$ and $K^2$ and revising the time domain.

Repeat the above procedures for job $K^i$, which is the job with the smallest execution time among the un-preempted jobs in schedule $S_{K_{i+1}}$, for $i = 2, 3, \ldots, \lceil K \rceil - 1$. We could define the job sets $K^i$ and $K_2^i$, for $i = 2, 3, \ldots, \lceil K \rceil - 1$. For the brevity, let job sets $K_1^0, K_1^2$ represent job sets $K_1, K_2$, respectively. We would show that the following properties hold for any $i = 0, 1, \ldots, \lceil K \rceil - 1$.

(Q1). Job sets $K_2^0$ and $K_2^1$ are both schedulable.

(Q2). For any job $K_j$ in job set $K_2^i$, the job set \( \{ K_j \} \cup \{ K_i \mid \} \) is not schedulable. (This property is similar to Lemma 1.)

(Q3). The execution time of any job in job set $K_2^i$ is no less than the execution time of job $K_i$ if $K_2^i$ is not an empty set. (This property is similar to Lemma 2.)

By definitions, we know that both job sets $K_2^0$ and $K_2^1$ are schedulable. Therefore, with Lemmas 1 and 2, we know that the properties (Q1), (Q2), and (Q3) hold when $i = 0$. We prove these properties by induction.

**Lemma 3:** Suppose that the properties (Q1), (Q2), and (Q3) hold when $i = 0, \ldots, m - 1$. Then, the properties (Q1), (Q2), and (Q3) also hold when $i = m$.

**Proof:** The proofs of the properties (Q2) and (Q3) could be obtained by very similar arguments in the proofs of Lemmas 1 and 2 when job sets $K_2^0$ and $K_2^1$ are schedulable, and the properties (Q2) and (Q3) hold for $i = 0, \ldots, m - 1$. We focus on proving (Q1). The schedulability is obvious for job set $K_2^0$, since we change the time domain according to the un-preempted job $K^m$ only. We focus our discussions on showing the schedulability of job set $K_2^m$. Let $s^m$ and $c^m$ be the starting time and completion time of job $K^m$ in the EDF schedule of job set $K_2^m$. Let $s^m$ and $c^m$ be the starting time and completion time of job $K^i$ in the EDF schedule of job set $K_i^m$ respectively. By definitions, the EDF schedule of job set $K_i^m$ completes all the jobs in job set $K_i^m$ in time. If both jobs $K^m$ and $K^m$ are empty, we know that no job is executed in time interval $(s^m, c^m)$ in the EDF schedule of job set $K_2^m$ because of the property (Q3) for $i = m - 1$. Similarly, if $K^m$ and $K^m$ indicate the same job, the schedulability of job set $K_2^i$ is obvious since no job would preempt job $K^m$ in time interval $(s^m, c^m)$ based on the property (Q3) when $i = m - 1$. We focus on the other case.

If job $K^m$ is not empty and job $K^m$ is empty, we know that all of the jobs, distinct from job $K^m$, executed in time interval $(s^m, c^m)$ in schedule $S_{K_i^m}$, are preempted by job $K^m$ based on the EDF property (P3) and their executions before the starting time of job $K^m$ in schedule $S_{K_i^m}$, Suppose that job $K^m$ completes at the time instant $c^m$ in schedule $S_{K_i^m}$, Therefore, $c^m$ is no greater than $c^m$. We know that job $K^m$ executes $e^m - (c^m - s^m)$ amount of execution time before $s^m$, where $e^m$ is the execution time of job $K^m$. Suppose that the amount of execution time of the other jobs executed in time interval $(c^m, c^m)$ in schedule $S_{K_i^m}$ is $x$. We know that $x + (c^m - s^m) \leq e^m - s^m \leq e^m = (c^m - s^m) + e^m - (c^m - s^m)$, where the second inequality comes from the property (Q3). Therefore, $x \leq e^m - (c^m - s^m)$. By advancing the execution of the jobs executed in time interval $(c^m, c^m)$ to the time intervals to execute job $K^m$ before $s^m$ in schedule $S_{K_i^m}$, we can derive a schedule to complete all of the jobs in job set $K_2^m \setminus \{ K^m \}$ in time without executing any job in time interval $(s^m, c^m)$. This is because schedule $S_{K_i^m}$ is feasible and no job starts its execution earlier than its original starting time or completes later than its original completion time. Therefore, job set $K_2^m$ is schedulable for such a case. Another case when job $K^m$ is empty and job $K^m$ is not empty is similar.

If $K^m \neq K^m \neq \phi$, there are three cases. For the first case, job $K^m$ completes before job $K^m$ starts in schedule $S_{K_i^m}$, The proof could be done by combining the analysis of the case when either $K^m$ or $K^m$ is empty. We consider the second case when job $K^m$ starts before job $K^m$ starts in schedule $S_{K_i^m}$, i.e., $K^m$ preempts $K^m$. For such a case, the completion time $c^m$ of job $K^m$ is less than that of job $K^m$, where $c^m < c^m$. It is not difficult to see that all of the jobs executed in time interval $(c^m, c^m)$ in schedule $S_{K_i^m}$ are preempted by job $K^m$. Advancing the jobs, excluding job $K^m$, executed in time interval $(c^m, c^m)$ to the time intervals for the executions of job $K^m$ in job set $K_2^m$ before $s^m$ and removing the executions of jobs $K^m$ and $K^m$ would derive a schedule to complete all of the jobs in job set $K_2^m \setminus \{ K^m, K^m \}$ in time without executing any job in time interval $(s^m, c^m)$. Similar arguments could also be made for the case when the completion time of job $K^m$ is greater than that of job $K^m$.

**Theorem 2:** Algorithm LEF is a 3-approximation algorithm for the IJS problem.

**Proof:** It suffices to prove \( \lceil J \rceil \geq 2 \lceil J \rceil \) by showing that at most two jobs are in job set $K_2^1$ since we remove at most two jobs to construct job set $K_2^1$ from job set $K_2^0$ for $i = 1, 2, \ldots, \lceil J \rceil - 1$. By the properties (Q2) and (Q3), we know that the EDF schedule of job set $K_2^1$ could only schedule at most two jobs, one starts before the start time of the feasible interval of the unique job $K_2^1$ in $K_2^1$, and another starts before the start time of the feasible interval of job $K_2^1$ plus the execution time of $K_2^1$. With Lemmas 1 and 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 $J = \{ J_1, J_2, J_3 \}$, where $J_1 = (b, \{ b, 2b, 3b, 4b \}), J_2 = (b + 5, \{ b + 5, 2b - 5, 3b \})$, and $J_3 = (b + 5, \{ b + 5, 2b - 5, 3b \}) (b > 0$ and $0.5b \geq e > 0$). The optimal schedule is to execute $J_1, J_3,$ and $J_2$ in their second, first and first feasible intervals, respectively, whereas the derived schedule of Algorithm LEF only executes $J_1$ in its first feasible interval.

5. Performance Evaluation

The experiments described in this section are to access the capability of our algorithms in scheduling jobs with multiple feasible intervals. For preemptible jobs, we compare the performance of Algorithm LEF against those algorithms in [8], including Fewer Feasible Interval First-based (FFFF-First Fit, FFF-Last Fit, FFF-Worst Fit, FFF-Best Fit) and First Come First Serve-based algorithms (FCFS-First Fit, FCFS-Last Fit, FCFS-Worst Fit, and FCFS-Best Fit). For non-preemptible jobs, we compare Algorithm LEF against 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.

A. Workload Generation

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type I</th>
<th>Type II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of jobs</td>
<td>3, 10, 12, 14, 16, 18</td>
<td>3, 10, 12, 14, 16, 18</td>
</tr>
<tr>
<td>Average release rate per second</td>
<td>$\mathcal{E}(\text{Poisson})$</td>
<td>$\mathcal{E}(\text{Poisson})$</td>
</tr>
<tr>
<td>Execution time</td>
<td>uniform(200, 600)µs</td>
<td>uniform(200, 600)µs</td>
</tr>
<tr>
<td>Number of feasible intervals</td>
<td>uniform(2, 3)</td>
<td>uniform(2, 3)</td>
</tr>
<tr>
<td>Interval length</td>
<td>uniform(200, e) , 500µs</td>
<td>uniform(200, e) , 500µs</td>
</tr>
<tr>
<td>Temporal distance</td>
<td>uniform(200, 500)µs</td>
<td>uniform(200, 500)µs</td>
</tr>
</tbody>
</table>

TABLE I: Simulation Parameters for Type I and Type II, where $e$ denotes the execution time of the job determined.
The job sets in our simulations are generated based on two parameters: the number of jobs and average release rate. The former is the number of jobs in the job set; the latter is the average number of jobs released within each second. 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 $i$, of job $j_i$ and the end time of feasible interval $j_{i-1}$, of job $j_i$, for $j > 1$. Two types of workload with parameter settings listed in Table I, referred to as Type I and Type II, are considered in the simulations. Experimental results are conducted from 512 independent experiments for each parameter configuration.

We use completion rate and normalized completion rate for performance measurements. The completion rate of an algorithm for an input instance is defined as the ratio of the number of jobs completed in time in the derived schedule to that of the optimal solution. The normalized completion rate 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. The measurements on completion rates of the simulated algorithms provide performance indexes for the cases that the optimal solution could be derived efficiently, while the measurements on normalized completion rates are for approximated performance indexes for the other cases.

### B. Experimental Results

For preemptible jobs, the simulation results are presented in Figures 2(a) and 2(b) for workloads generated from Type I and Type II by measuring the average completion rate and the average normalized completion rate, respectively. In Figure 2(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 release time of jobs which are not highly related factors. For workloads generated from Type I, 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 strategies do. Furthermore, the performance of Algorithm LEF is more steady in Figure 2(a), in which the average completion rates of FFIF-based algorithms and FCFS-based algorithms decrease when the number of jobs increases. In Figure 2(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 Figure 2(a) are similar to those in Figure 2(b).

For non-preemptible jobs, the simulation results are presented in Figures 3(a) and 3(b) for workloads generated from Type I and Type II by measuring the average completion rate and the average normalized completion rate, respectively. The average completion rates for Algorithm LEF and Algorithm FCF range from 87% to 90% and 77% to 80%, respectively. The average normalized completion rates for Algorithm LEF and Algorithm FCF range from 81% to 83% and 75% to 78%, respectively.

### 6. Conclusion

In this paper, we present two approximation algorithms for multiple feasible interval jobs. For preemptible jobs, Algorithm LEF 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 heuristics to choose the preemptible job with the least execution time, Algorithm LEF is proved being a 3-approximation algorithm for 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 LEF and Algorithm LEF could complete more jobs in time than the previous heuristic algorithms could.

### References


