Online Flexible Job Scheduling for Minimum Span

Runtian Ren
School of Computer Science and Engineering
Nanyang Technological University
Singapore 639798
RENR0002@e.ntu.edu.sg

Xueyan Tang
School of Computer Science and Engineering
Nanyang Technological University
Singapore 639798
asxytang@ntu.edu.sg

ABSTRACT

In this paper, we study an online Flexible Job Scheduling (FJS) problem. The input of the problem is a set of jobs, each having an arrival time, a starting deadline and a processing length. Each job has to be started by the scheduler between its arrival and its starting deadline. Once started, the job runs for a period of the processing length without interruption. The target is to minimize the span of all the jobs — the time duration in which at least one job is running. We study online FJS under both the non-clairvoyant and clairvoyant settings. In the non-clairvoyant setting, the processing length of each job is not known for scheduling purposes. We first establish a lower bound of μ on the competitive ratio of any deterministic online scheduler, where μ is the max/min job processing length ratio. Then, we propose two $O(\mu)$-competitive schedulers: *Batch* and *Batch+*. The Batch+ scheduler is proved to have a tight competitive ratio of $(\mu + 1)$. In the clairvoyant setting, the processing length of each job is known at its arrival and can be used for scheduling purposes. We establish a lower bound of $\frac{\sqrt{5}+1}{2}$ on the competitive ratio of any deterministic online scheduler, and propose two $O(1)$-competitive schedulers: *Classify-by-Duration Batch+* and *Profit*. The Profit scheduler can achieve a competitive ratio of $4+2\sqrt{2}$. Our work lays the foundation for extending several online job scheduling problems in cloud and energy-efficient computing to jobs that have laxity in starting.

KEYWORDS

Job scheduling, span, online algorithm

ACM Reference format:
Runtian Ren and Xueyan Tang. 2017. Online Flexible Job Scheduling for Minimum Span. In *Proceedings of SPAA '17, July 24-26, 2017, Washington DC, USA, ,* 12 pages.
DOI: http://dx.doi.org/10.1145/3087556.3087562

1 INTRODUCTION

We study the following scheduling problem: the input is a set of jobs, each having an arrival time, a starting deadline and a processing length. Each job has to be started by the scheduler (algorithm) between its arrival and its starting deadline. Once started, the job runs for a period of the processing length without interruption. The target is to minimize the span of all the jobs, where the span is the time duration in which at least one job is running. We refer to this problem as the *Flexible Job Scheduling* (FJS) problem. In this paper, we consider the online version of FJS, where the scheduler does not have any knowledge of future job arrivals.

In a sense, the objective of FJS is to maximize the parallelism of job execution — a recent trend arising from the prevalence of cloud computing and energy-efficient computing. If the execution of jobs can always be hosted by one server with a sufficient capacity, minimizing the span directly minimizes the running hours of the server. Under the pay-as-you-go billing model of clouds, this minimizes the monetary expense [1]. In energy-efficient computing, this indicates reducing the energy consumption of the server as the power usage of a server normally consists of a component proportional to the time duration when the server is "on" (which represents the power consumed by an idle server) and a component proportional to the total amount of work done (which is fixed for processing a given set of jobs) [4]. If job execution needs to be distributed to multiple servers each with a limited capacity, minimizing the total running hours of the servers can be modeled as the MinUsageTime Dynamic Bin Packing problem [15, 16, 19], for which the competitiveness of scheduling often depends on the span and the time-accumulated resource demand of all the jobs. Thus, minimizing the span also helps enhance the competitiveness of scheduling.

FJS can be seen as a variant of the interval scheduling problem [13, 14]. In interval scheduling, a set of jobs are to be scheduled on machines. After a job arrives, it needs to run for its processing length on a machine non-preemptively and be completed before a deadline. In many cases, each machine can run only one job at a time. A common target of interval scheduling is to schedule a subset of jobs on a given number of machines such that the total weight of the scheduled jobs is maximized [2, 3, 5, 8, 9, 17, 21, 24]. This objective is substantially different from ours in that it attempts to control the parallelism of job execution. Recently, another version of interval scheduling closer to our problem is receiving increasing attention [7, 10, 11, 18, 22]. In this version, each machine can run a fixed number of jobs concurrently. The target is to minimize the total busy time of all the machines used for scheduling all the jobs. Most work along this direction has focused on scheduling jobs in an offline manner, where all the inputs are given in advance. There was only one study on online scheduling for optimizing the busy time [22]. However, it assumed "rigid" jobs that must be started immediately at their arrivals (i.e., each job's deadline is exactly its arrival time plus its processing length). Different from [22], we consider flexible jobs that have laxity in starting. A recent work showed that the offline version of FJS can be solved by dynamic programming [11], but no online scheduler was investigated. Our

SPAA '17, July 24-26, 2017, Washington DC, USA

DOI: http://dx.doi.org/10.1145/3087556.3087562

work focuses on the online version of FJS and lays the foundation for extending online busy time optimization to flexible jobs.

Contributions of this paper. We study online FJS under both the non-clairvoyant and clairvoyant settings. In the non-clairvoyant setting, the processing length of each job is not known until it completes execution. Thus, the knowledge of processing length cannot be used for scheduling purposes. We establish a lower bound of μ on the competitive ratio of any deterministic online scheduler, where μ is the max/min job processing length ratio. Then, we propose two $O(\mu)$-competitive schedulers: *Batch* and *Batch+*. The Batch scheduler achieves a competitive ratio between 2μ and $(2\mu + 1)$, and the Batch+ scheduler achieves a tight competitive ratio of $(\mu + 1)$. In the clairvoyant setting, the processing length of each job is known at the time of its arrival and this information can be used by the scheduler. We establish a lower bound of $\frac{\sqrt{5}+1}{2}$ on the competitive ratio of any deterministic online scheduler. Then, we propose two $O(1)$-competitive schedulers: *Classify-by-Duration Batch+* and *Profit*. We show that these two schedulers can achieve competitive ratios of $7 + 2\sqrt{6}$ and $4 + 2\sqrt{2}$ respectively.

2 PRELIMINARIES

We first define some key notations for this paper. For any time interval I, we use I^- and I^+ to denote the left and right endpoints of I respectively. For technical reasons, we shall view intervals as half-open, i.e., $I = [I^-, I^+)$. Let $\text{len}(I) = I^+ - I^-$ denote the length of interval I.

For any job J, let $a(J)$, $d(J)$ and $p(J)$ denote J's arrival time, starting deadline (latest possible starting time)[1] and processing length respectively. When a job J arrives at time $a(J)$, its starting deadline $d(J)$ is known and the job can be started at any time between $a(J)$ and $d(J)$. The difference $d(J) - a(J)$ is known as the *laxity* of job J. Once the job is started, it has to run for a processing length $p(J)$ without interruption. A *scheduler* decides the starting times of jobs. If a scheduler Λ starts a job J at time t, we refer to the time interval $\Big[t, t + p(J)\Big)$ in which J is running as the *active interval* of job J and denote it by $I_\Lambda(J)$. Let $\mathcal{J}$ denote the set of jobs to schedule. The time duration in which at least one job in $\mathcal{J}$ is running is known as the *span* of $\mathcal{J}$. We denote the span of $\mathcal{J}$ induced by a scheduler Λ as $\text{span}_\Lambda(\mathcal{J}) = \text{len}\Big(\bigcup_{J\in\mathcal{J}} I_\Lambda(J)\Big)$.

Our goal of job scheduling is to minimize the span. We denote the minimum possible span induced by an optimal scheduler as $\text{span}_{min}(\mathcal{J})$. The performance of an online algorithm is often characterized by its *competitive ratio*, i.e., the worst-case ratio between the objective function values of an online solution and an optimal solution [6]. We propose several online schedulers and analyze their competitiveness.

3 NON-CLAIRVOYANT SETTING

We start by considering the non-clairvoyant setting first. In Section 3.1, we show that any deterministic online scheduler has a competitive ratio at least μ, where $\mu = \frac{\max_{J\in\mathcal{J}} p(J)}{\min_{J\in\mathcal{J}} p(J)}$ is the ratio of the maximum job processing length to the minimum job processing length among all the jobs to schedule. In Section 3.2, we introduce a $(2\mu + 1)$-competitive *Batch* scheduler and a $(\mu + 1)$-competitive *Batch+* scheduler.

[1]We assume a starting deadline for each job rather then a completion deadline as one of the settings we study is non-clairvoyant where the processing length of each job is not known.

3.1 Lower Bound on Competitiveness of Online Schedulers

We construct an instance to establish the lower bound μ on the competitive ratio of any deterministic online scheduler. In our instance, each job has one of two processing lengths: 1 or μ, where $\mu > 1$ can be revealed to the scheduler in advance. In the non-clairvoyant setting, an adversary has the freedom of assigning a processing length to a job after it is started. In our instance, each job is assigned the processing length 1 time unit after it is started since the processing lengths of all jobs are at least 1. If a job J is assigned a processing length of 1, it completes execution immediately. If the job is assigned a processing length of μ, it continues to run for another $(\mu - 1)$ time units before completing execution. Let $k \geq 1$ be an integer. We release jobs in up to $(k + 1)$ iterations. For each iteration i, let T_i denote the time to release all the jobs of this iteration. Among all the jobs released in iteration i, the number of concurrently running jobs induced by an online scheduler shall be referred to as the *concurrency* of iteration i. Note that each iteration has its own concurrency and the concurrency normally changes over time.

In the first iteration, we start by releasing $2^{2^{2k}}$ jobs at time $T_1 = 0$. Let the j-th job ($1 \leq j \leq 2^{2^{2k}}$) have a laxity of α^j, where $\alpha > \mu + 1$ is a constant. As long as the concurrency of the first iteration does not exceed $\sqrt{2^{2^{2k}}} = 2^{2^{2k-1}}$ by applying the online scheduler, all the jobs are assigned a processing length of 1 when they are due for the length assignment. If the concurrency never exceeds $2^{2^{2k-1}}$ until all the jobs complete execution, we stop releasing jobs and do not proceed with any further iteration. Otherwise, consider the time t_1 when the concurrency first exceeds $2^{2^{2k-1}}$. Suppose there are m_1 ($m_1 \geq 2^{2^{2k-1}} + 1$) jobs running at time t_1. Let $J_1, J_2, \ldots, J_{m_1}$ be all these jobs sorted in an increasing order of their laxities. Note that none of these jobs has been assigned the processing length, since all the jobs assigned the processing length so far were assigned a length of 1 and have completed execution. We shall assign a processing length of μ to job J_{m_1} and refer to it as the *earmarked job* of the first iteration. All the other jobs $J_1, J_2, \ldots, J_{m_1-1}$ will be assigned the same processing length of 1. Apart from this, all the jobs started after time t_1 will also be assigned a processing length of 1. Then, we set T_2, the job release time of the second iteration, to the completion time of the earmarked job J_{m_1} of the first iteration.

The above process is repeated iteratively with decreasing number of jobs released in each iteration. In each iteration i ($2 \leq i \leq k$), we release $2^{2^{2k-i+1}}$ jobs at time T_i. Let the j-th job ($1 \leq j \leq 2^{2^{2k-i+1}}$) have a laxity of α^j. As long as the concurrency of iteration i does not exceed $\sqrt{2^{2^{2k-i+1}}} = 2^{2^{2k-i}}$ by applying the online scheduler, we assign a processing length of 1 to all the jobs. If the concurrency never exceeds $2^{2^{2k-i}}$ until all the jobs released in iteration i complete execution, we do not proceed with any further iteration of job release. Otherwise, let t_i denote the time when the concurrency of iteration i first exceeds $2^{2^{2k-i}}$. Suppose there are m_i ($m_i \geq 2^{2^{2k-i}}+1$) jobs running at time t_i, and let $J_1, J_2, \ldots, J_{m_i}$ be all these jobs sorted

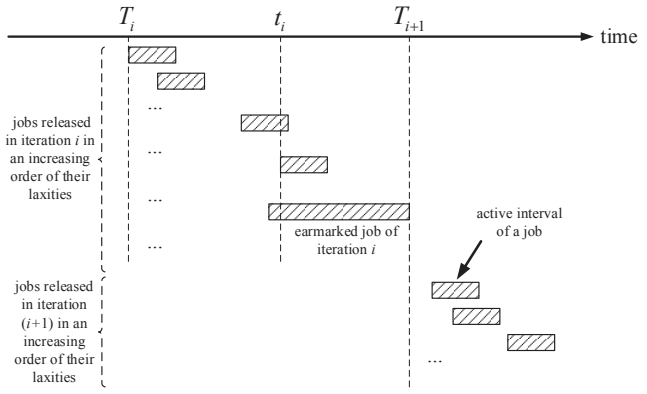

Figure 1: Jobs released in iterations i and $(i+1)$

in an increasing order of their laxities. We shall assign a processing length of μ to job J_{m_i} and refer to it as the *earmarked job* of iteration i. All the other jobs $J_1, J_2, \ldots, J_{m_i-1}$ will be assigned the same processing length of 1. In addition, we shall assign a processing length of 1 to all the jobs started after time t_i. Then, we set T_{i+1}, the job release time of iteration $(i+1)$, to the completion time of the earmarked job J_{m_i} of iteration i. Figure 1 illustrates the above process. In the final iteration $(k+1)$ (if needed), we release 2^{2^k} jobs at time T_{k+1} and directly assign the same processing length of 1 to all these jobs.

We introduce two lemmas for analyzing the online scheduler and the optimal scheduler of the above instance.

Lemma 3.1. *For each iteration i $(1 \le i \le k)$, if there is no earmarked job, the span induced by the online scheduler for all the jobs released in this iteration must be at least $2^{2^{2k-i}}$.*

Proof: An iteration i does not have an earmarked job only if its concurrency never exceeds $2^{2^{2k-i}}$. In this case, all the jobs released in iteration i have the same processing length of 1, so their total length is $2^{2^{2k-i+1}}$. Thus, the span of these jobs is at least $2^{2^{2k-i+1}} \Big/ 2^{2^{2k-i}} = 2^{2^{2k-i}}$. □

Lemma 3.2. *For each iteration i $(2 \le i \le k+1)$, all the earmarked jobs of the first $(i-1)$ iterations can be started at the job release time T_i of iteration i.*

Proof: For each iteration j $(1 \le j < i)$, let t_j denote the time when the concurrency of iteration i first exceeds $2^{2^{2k-j}}$. Among all the m_j ($m_j \ge 2^{2^{2k-j}} + 1$) jobs active at t_j, let $\widetilde{J}_j$ be the job with the smallest laxity and let $\widehat{J}_j$ be the job with the largest laxity ($\widehat{J}_j$ is the earmarked job of iteration j). Denote the laxity of job $\widehat{J}_j$ by α^{L_j}. Since the jobs released in iteration j have exponentially increasing laxities, it is easy to infer that $L_j \ge m_j \ge 2^{2^{2k-j}} + 1$ and the laxity of job $\widetilde{J}_j$ is at most $\alpha^{L_j - m_j + 1}$. We next show that $T_i \le T_j + \alpha^{L_j}$ for each $1 \le j < i$, which suggests that the earmarked job of iteration j can be started at time T_i.

Note that in iteration j, the earmarked job $\widehat{J}_j$ is active at time t_j and has a processing length of μ. Since T_{j+1} is exactly the completion time of job $\widehat{J}_j$, we have $T_{j+1} \le t_j + \mu$. On the other hand, job $\widetilde{J}_j$ is also active at time t_j and has a processing length of 1. Thus, $\widetilde{J}_j$ must be started no earlier than $t_j - 1$. Since the laxity of $\widetilde{J}_j$ is at most $\alpha^{L_j - m_j + 1}$, $\widetilde{J}_j$'s starting deadline is bounded by $T_j + \alpha^{L_j - m_j + 1}$. As a result, we have $t_j - 1 \le T_j + \alpha^{L_j - m_j + 1}$. Therefore,

$$T_{j+1} \le t_j + \mu \le T_j + (\mu + 1) + \alpha^{L_j - m_j + 1}.$$

Since the above inequality holds for each $1 \le j < i$. It follows that

$$T_i \le T_j + (i-j)(\mu+1) + \sum_{l=j}^{i-1} \alpha^{L_l - m_l + 1}.$$

Now, to prove $T_i \le T_j + \alpha^{L_j}$, we only need to show $(i-j)(\mu+1) + \sum_{l=j}^{i-1} \alpha^{L_l - m_l + 1} \le \alpha^{L_j}$.

For each $j \le l < i$, we have $L_l \ge m_l \ge 2^{2^{2k-l}} + 1$. Since the total number of jobs released in iteration $(l+1)$ is $2^{2^{2k-l}}$, we also have $2^{2^{2k-l}} \ge L_{i+1}$. Thus, $L_l \ge L_{l+1} + 1$ for each $j \le l < i$. Consequently, $L_j \ge L_l + (l-j)$. Note that $m_l \ge 3$. Therefore,

$$\begin{aligned}
\sum_{l=j}^{i-1} \alpha^{L_l - m_l + 1} &= \left(\sum_{l=j+1}^{i-1} \alpha^{L_l - m_l + 1} \right) + \alpha^{L_j - m_j + 1} \\
&< \left(\sum_{l=j+1}^{i-1} \alpha^{L_l - 2} \right) + \frac{\alpha^{L_j}}{\alpha^2} \\
&\le \left(\frac{\alpha^{L_j}}{\alpha^3} + \frac{\alpha^{L_j}}{\alpha^4} + \cdots + \frac{\alpha^{L_j}}{\alpha^{i-j+1}} \right) + \frac{\alpha^{L_j}}{\alpha^2} \\
&< \left(\sum_{h=2}^{\infty} \frac{1}{\alpha^h} \right) \cdot \alpha^{L_j}
\end{aligned}$$

Since $\alpha > \mu + 1 \ge 2$ and $i - j \le k$, we have $(i-j)(\mu+1) < k\alpha$. It is easy to verify that when $\alpha > 2$, $k\alpha < \alpha^{2^k}$ for any $k \ge 1$. Since $L_j \ge m_j \ge 2^{2^{2k-j}} + 1 \ge 2^{2^k} + 1 > 2^k + 1$, we have $\alpha^{2^k} < \frac{1}{\alpha} \cdot \alpha^{L_j}$. As a result, $(i-j)(\mu+1) < \frac{1}{\alpha} \cdot \alpha^{L_j}$. Thus,

$$(i-j)(\mu+1) + \sum_{l=j}^{i-1} \alpha^{L_l - m_l + 1} < \left(\sum_{h=1}^{\infty} \frac{1}{\alpha^h} \right) \cdot \alpha^{L_j} = \frac{\alpha^{L_j}}{\alpha - 1} < \alpha^{L_j},$$

where the last inequality holds since $\alpha > 2$. Therefore, $T_i \le T_j + \alpha^{L_j}$ for each $1 \le j < i$ and the lemma is proven. □

Now, we derive the ratio between the spans induced by the online scheduler and the optimal scheduler. In the first iteration, if there is no earmarked job, according to Lemma 3.1, the span induced by the online scheduler is at least $2^{2^{2k-1}}$. In this case, the optimal schedule is to start all the jobs immediately once they are released (i.e., at time $T_1 = 0$) so that the induced span is 1. Thus, the ratio between the two spans is at least $2^{2^{2k-1}}$. If there is an earmarked job in the first iteration, the second iteration follows.

If the job releasing process is terminated after i iterations ($2 \le i \le k$), each of the first $(i-1)$ iterations has an earmarked job. Since the jobs of each iteration are released when the earmarked job of the previous iteration completes, the active intervals of these $(i-1)$ earmarked jobs do not overlap. Moreover, by Lemma 3.1, the span induced by the online scheduler for the jobs released in iteration i is at least $2^{2^{2k-i}}$. Thus, the online scheduler must induce a total span at least $(i-1)\mu + 2^{2^{2k-i}}$. A better schedule is to start all the jobs released in iteration i together with all earmarked jobs of the

first $(i-1)$ iterations at time T_i (Lemma 3.2). All the remaining jobs released in the first $(i-1)$ iterations (which have processing lengths of 1) can be started immediately once they are released. In this way, the total span is $\mu + (i-1)$. Therefore, the ratio between the spans induced by the online scheduler and the optimal scheduler is at least $\frac{(i-1)\mu+2^{2^{2k-i}}}{\mu+(i-1)}$.

Finally, if the job releasing process proceeds to iteration $(k+1)$, the span induced by the online scheduler must be at least $k\mu+1$. A better schedule is to start all the jobs released in iteration $(k+1)$ together with all the earmarked jobs of the previous iterations at time T_{k+1} (Lemma 3.2). All the remaining jobs are started immediately once they are released. This schedule induces a span at most $\mu + k$. So, the ratio between the two spans is at least $\frac{k\mu+1}{\mu+k}$.

In summary, the competitive ratio by applying any deterministic online scheduler is at least

$$\min\left\{2^{2^{2k-1}}, \min_{2\le i\le k}\left\{\frac{(i-1)\mu+2^{2^{2k-i}}}{\mu+(i-1)}\right\}, \frac{k\mu+1}{\mu+k}\right\}.$$

Since $\frac{(i-1)\mu+2^{2^{2k-i}}}{\mu+(i-1)} > \frac{2^{2^k}}{\mu+k}$ and $\frac{k\mu+1}{\mu+k} > \frac{\mu}{\frac{\mu}{k}+1}$, the above competitive ratio can be made arbitrarily close to μ as k goes towards infinity. Thus, we can conclude that:

Theorem 3.3. *For Non-Clairvoyant FJS, no deterministic online scheduler can achieve a competitive ratio less than μ, where μ is the max/min job processing length ratio.*

3.2 Batch Scheduler and Batch+ Scheduler

It is easy to infer that an eager scheduler that starts every job immediately at its arrival cannot achieve any bounded competitive ratio even for any given μ, because it does not make use of any laxity to boost the concurrency of job execution. Similarly, a lazy scheduler that delays the start of each job till its starting deadline cannot achieve any bounded competitive ratio for any given μ either, since it does not take any advantage of the flexibility offered by the laxity. Without any knowledge of job processing length, an intuitive strategy to improve the concurrency of job execution is to start as many jobs as possible simultaneously. In the following, we first propose a *Batch* scheduler and show that it can achieve a competitive ratio of $(2\mu+1)$ for Non-Clairvoyant FJS.

The Batch scheduler determines the starting times of jobs in iterations. In each iteration, the Batch scheduler waits until a job J pending execution hits its starting deadline $d(J)$, i.e., J is the job with the earliest starting deadline among those that have arrived but not yet started execution. We refer to J as the *flag job* of the iteration. If several jobs share the same starting deadline, any one of them can be chosen as the flag job. At time $d(J)$, the Batch scheduler starts all the jobs pending execution and let them run concurrently. Then, the scheduler proceeds to the next iteration to wait for another job to hit its starting deadline and repeats this process.

Theorem 3.4. *For Non-Clairvoyant FJS, the Batch scheduler achieves a competitive ratio between 2μ and $(2\mu+1)$, where μ is the max/min job processing length ratio.*

Proof: Let J_i denote the flag job of iteration i as designated by the Batch scheduler. By definition, the flag jobs $J_1, J_2, J_3, \ldots$ have increasing starting deadlines. To establish a lower bound on the span induced by an optimal scheduler, we choose a subset of the flag jobs, such that their active intervals cannot overlap with each other by any scheduler. First, we choose J_1. Then, we find the lowest-indexed flag job J_h that satisfies $d(J_h) \ge d(J_1) + p(J_1)$ and choose the flag job J_{h+1} if J_{h+1} exists. This selection process is repeated. After a flag job J_i is chosen, we find the lowest-indexed flag job J_j that satisfies $d(J_j) \ge d(J_i) + p(J_i)$ and then choose the flag job J_{j+1} if J_{j+1} exists. By the definition of Batch, J_{j+1} arrives later than $d(J_i)$ and thus, $a(J_{j+1}) > d(J_j) \ge d(J_i) + p(J_i)$. Note that $d(J_i) + p(J_i)$ is the latest possible completion time of job J_i. This implies that the active intervals of J_{j+1} and J_i cannot overlap with each other by any scheduler. Thus, the span induced by an optimal scheduler is at least $\sum_{J\in\mathcal{F}} p(J)$, where $\mathcal{F}$ is the subset of flag jobs chosen.

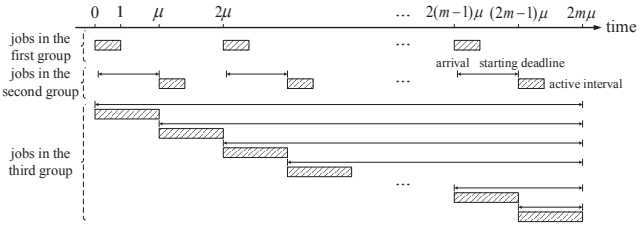

Figure 2: An example schedule induced by Batch

Next, we derive an upper bound on the span induced by the Batch scheduler. Consider any two flag jobs J_i and J_{i+1} chosen in succession. Since J_j is the lowest-indexed flag job satisfying $d(J_j) \ge d(J_i) + p(J_i)$, for each flag job J_g $(i \le g < j)$, it holds that $d(J_g) < d(J_i) + p(J_i)$. This means that all the jobs started in iterations i to $(j-1)$ by the Batch scheduler are started before time $d(J_i) + p(J_i)$. Thus, all these jobs must complete execution by time $d(J_i) + p(J_i) + \mu \cdot p(J_i)$, since the max/min job processing length ratio is μ. Therefore, the span induced by the Batch scheduler for iterations i to $(j-1)$ is bounded by $d(J_i) + p(J_i) + \mu \cdot p(J_i) - d(J_i) = (\mu+1) \cdot p(J_i)$. Similarly, for all the jobs started in iteration j, since they are started simultaneously, their induced span is bounded by the maximum job processing length, which is in turn capped by $\mu \cdot p(J_i)$. Consequently, the span induced by the Batch scheduler for iterations i to j is bounded by $(\mu+1)\cdot p(J_i) + \mu \cdot p(J_i) = (2\mu+1)\cdot p(J_i)$. Putting all the iterations together, the total span induced by the Batch scheduler is bounded by $(2\mu+1) \cdot \sum_{J\in\mathcal{F}} p(J)$, where $\mathcal{F}$ is the subset of flag jobs chosen. Therefore, the competitive ratio of the Batch scheduler is bounded by $(2\mu+1)$.

Now, we construct an instance to show that the competitive ratio of the Batch scheduler is no less than 2μ. Consider three groups of jobs as shown in Figure 2. The first group contains m short jobs, each with a laxity of 0 and a processing length of 1. The i-th short job in the first group arrives at time $2(i-1)\mu$. The second group also contains m short jobs, each with a laxity of $(\mu-\varepsilon)$ and a processing length of 1, where $\varepsilon > 0$ can be arbitrarily small. The i-th short job in the second group arrives at time $2(i-1)\mu + \varepsilon$. The third group contains $2m$ long jobs, all of which have the same starting deadline of $2m\mu$ and a processing length of μ. The i-th long job arrives at time $(i-1)\mu$. By applying the Batch scheduler, the i-th short job in the first group and the $(2i-1)$-th long job in the third group are started together at time $2(i-1)\mu$ in the same iteration. The i-th short job

in the second group and the $(2i)$-th long job in the third group are started together at time $(2i-1)\mu$ in the same iteration. Therefore, the total span induced is $2m\mu$. A better schedule is to start all the long jobs in the third group at their (same) starting deadlines and start all the short jobs in the first two groups immediately at their arrivals. This yields a span of $m(1+\varepsilon)+\mu$. Therefore, the ratio between the spans induced by the Batch scheduler and the optimal scheduler is at least $\frac{2m\mu}{m(1+\varepsilon)+\mu}$, which can be made arbitrarily close to 2μ as m goes towards infinity. □

To improve the competitiveness, we can make the Batch scheduler more aggressive. Next, we propose a tight $(\mu+1)$-competitive *Batch+* scheduler. Same as the Batch scheduler, Batch+ also determines the starting times of jobs in iterations. In each iteration, the Batch+ scheduler determines a flag job (the job with the earliest starting deadline) and starts all the pending jobs together with the flag job. What is different from Batch is that, during the active interval of the flag job, the Batch+ scheduler further starts all the newly arriving jobs immediately at their arrivals. Only when the flag job completes execution, the Batch+ scheduler starts the next iteration to buffer incoming jobs and search for a new flag job.

Theorem 3.5. *For Non-Clairvoyant FJS, the Batch+ scheduler achieves a tight competitive ratio of* $(\mu+1)$*, where* μ *is the max/min job processing length ratio.*

Proof: We first show that the Batch+ scheduler is $(\mu+1)$-competitive. Let J_i denote the flag job of iteration i as designated by the Batch+ scheduler. By applying Batch+, all the jobs started in iteration i are started no later than the completion time of J_i, i.e., $d(J_i)+p(J_i)$. Since the max/min job processing length ratio is μ, the span of the jobs started in iteration i is at most $(\mu+1)\cdot p(J_i)$. Therefore, the total span induced by the Batch+ scheduler is at most $(\mu+1)\cdot\sum_{i\geq 1} p(J_i)$. Note that the flag job J_{i+1} of iteration $(i+1)$ must arrive later than $d(J_i)+p(J_i)$. Thus, the active intervals of J_i and J_{i+1} cannot overlap with each other by any scheduler. Therefore, the span induced by an optimal scheduler is at least $\sum_{i\geq 1} p(J_i)$. As a result, the competitive ratio of the Batch+ scheduler is bounded by $(\mu+1)$.

We construct an instance to show that the competitive ratio of $(\mu+1)$ is tight. Consider two groups of jobs as shown in Figure 3. One group contains m short jobs, each with a laxity of 0 and a processing length of 1. The i-th short job arrives at time $(i-1)(\mu+1)$. The other group contains m long jobs, all of which have the same starting deadline of $m(\mu+1)$ and a processing length of μ. The i-th long job arrives at time $(i-1)(\mu+1)+(1-\varepsilon)$, where $\varepsilon>0$ can be arbitrarily small. By applying the Batch+ scheduler, the i-th short job and the i-th long job are started immediately at their arrivals in the same iteration, and thus a total span of $m(\mu+1-\varepsilon)$ is induced. An optimal schedule is to start all long jobs at their (same) starting deadlines and start all the short jobs immediately at their arrivals. This yields a span of $m+\mu$. Therefore, the ratio between the spans induced by the Batch+ scheduler and the optimal scheduler is $\frac{m(\mu+1-\varepsilon)}{m+\mu}$, which can be made arbitrarily close to $(\mu+1)$ as m goes towards infinity. □

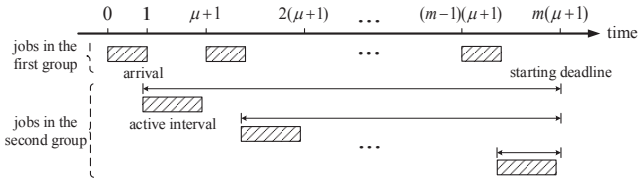

Figure 3: An example schedule induced by Batch+

4 CLAIRVOYANT SETTING

In this section, we study FJS under the clairvoyant setting, where the processing length of each job is known at its arrival. In Section 4.1, we establish a lower bound of $\frac{\sqrt{5}+1}{2}$ on the competitive ratio of any deterministic online scheduler. In Sections 4.2 and 4.3, we propose two $O(1)$-competitive schedulers: a *Classify-by-Duration Batch+* scheduler and a *Profit* scheduler.

4.1 Lower Bound on Competitiveness of Online Schedulers

We first construct an instance to show that any deterministic online scheduler has a competitive ratio of at least $\varphi=\frac{\sqrt{5}+1}{2}$. In our instance, each job has one of two processing lengths: 1 or φ. Let $n\geq 1$ be an integer. We release jobs in up to n iterations. In each iteration, we release two jobs, a short job with a processing length of 1 and a long job with a processing length of φ.

In the first iteration, we release a short job J_1 and a long job J_2 at time 0. Job J_1 has a laxity of 0 and job J_2 has a laxity of $n(\varphi+1)$. Obviously, J_1 must be started immediately at time 0. We check whether the online scheduler starts J_2 during the active interval of J_1, i.e., $[0,1)$. If J_2 is not started within J_1's active interval, we terminate the job releasing process and do not proceed with any further iteration. In this case, the span induced by the online scheduler is $\varphi+1$. The optimal schedule is to start J_1 and J_2 together at time 0, which induces a span of φ. Thus, the ratio between the two spans is $\frac{\varphi+1}{\varphi}=\varphi$. Otherwise, if J_2 is started during the time interval $[0,1)$, the second iteration follows.

The above process is repeated iteratively. As shown in Figure 4, in each iteration i $(2\leq i<n)$, we release a short job J_{2i-1} and a long job J_{2i} at time $(i-1)(\varphi+1)$. Job J_{2i-1} has a laxity of 0 and job J_{2i} has a laxity of $(n-i+1)(\varphi+1)$. We check whether the online scheduler starts J_{2i} during the active interval of J_{2i-1}, i.e., $[(i-1)(\varphi+1),(i-1)(\varphi+1)+1)$. If J_{2i} is not started within J_{2i-1}'s active interval, we terminate the job releasing process and do not proceed with any further iteration. In this case, the span induced by the online scheduler for iteration i is $(\varphi+1)$. Note that in each of the first $(i-1)$ iterations, the long job is started within the active interval of the short job. Since the job release times of two successive iterations are $(\varphi+1)$ apart, the active intervals of all the long jobs do not overlap with each other. Thus, the aggregate span of the first $(i-1)$ iterations is at least $(i-1)\varphi$. Therefore, the total span induced by the online scheduler for all the iterations is at least $(i-1)\varphi+(\varphi+1)$. An optimal schedule is to start all the long jobs together at time $(i-1)(\varphi+1)$ and to start all the short jobs at their respective arrivals. This yields a span of $\varphi+(i-1)$. Thus, the ratio between two spans is at least $\frac{(i-1)\varphi+\varphi+1}{\varphi+(i-1)}=\varphi$. Otherwise,

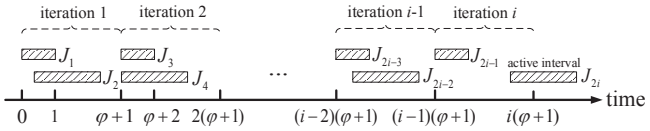

Figure 4: An example of the job release process

if J_{2i} is started during the active interval of J_{2i-1}, iteration $(i+1)$ follows.

In the final iteration n (if needed), we release a short job J_{2n-1} and a long job J_{2n} at time $(n-1)(\varphi+1)$. J_{2n-1} has a laxity of 0 and J_{2n} has a laxity of $(\varphi+1)$. Similar to the earlier argument, the active intervals of all the long jobs in the n iterations do not overlap with each other. Thus, the online scheduler induces a span at least $n\varphi$. An optimal schedule is to start all the long jobs at time $(n-1)(\varphi+1)$ and to start all the short jobs at their respective arrivals. This yields a span of $\varphi+(n-1)$. Thus, the ratio between the spans induced by the online scheduler and the optimal scheduler is at least $\frac{n\varphi}{\varphi+(n-1)}$, which can be made arbitrarily close to φ as n goes towards infinity.

With the above instance, we can conclude that:

Theorem 4.1. *For Clairvoyant FJS, no deterministic online scheduler can achieve a competitive ratio less than* $\frac{\sqrt{5}+1}{2}$.

4.2 Classify-by-Duration Batch+ Scheduler

As shown in Section 3.1, in the non-clairvoyant setting, the competitiveness of an online scheduler is limited by the max/min job processing length ratio μ. In the clairvoyant setting, with the knowledge of job processing length, a natural strategy to break this limit is to classify jobs into categories according to their lengths and schedule each category separately. In this way, the max/min job processing length ratio of each job category is reduced compared to the whole job set. In this section, we show that applying such a classify-by-duration strategy on the Batch+ scheduler can achieve an $O(1)$ competitive ratio.

We first introduce our *Classify-by-Duration Batch+* (CDB) scheduler. Let α be the max/min job processing length ratio of each job category. Given a base job processing length b, we classify jobs into categories such that each category includes all the jobs with processing length between $b\alpha^{i-1}$ and $b\alpha^{i}$ for an integer i. Whenever a new job arrives, it is put into a category according to its processing length. For each job category, we apply the Batch+ scheduler seperately to decide the starting times of jobs. If the max/min job processing length ratio for the entire set of jobs to schedule is μ, there can be up to $\lceil \log_\alpha \mu \rceil + 1$ non-empty job categories produced by the classification. Let $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_{\lceil \log_\alpha \mu \rceil+1}$ denote these job categories. Let $\mathcal{F}_i$ be the set of flag jobs designated by the Batch+ scheduler in the job category $\mathcal{J}_i$. Let $\mathcal{F} = \bigcup_{i=1}^{\lceil \log_\alpha \mu \rceil+1} \mathcal{F}_i$ be all the flag jobs designated by Classify-by-Duration Batch+ in the entire job set.

Lemma 4.2. *The span induced by the Classify-by-Duration Batch+ scheduler for a set of jobs* $\mathcal{J}$ *is bounded by* $(\alpha+1)$ *times the span of all the flag jobs* $\mathcal{F} \subseteq \mathcal{J}$ *in the schedule.*

Proof: By applying the Batch+ scheduler, the active interval of each flag job J is $\Big[d(J), d(J)+p(J)\Big)$. Thus, the span of all the flag jobs is $\bigcup_{J\in\mathcal{F}} \Big[d(J), d(J)+p(J)\Big)$. Such a span may not be a single contiguous time interval. It may be composed of multiple contiguous time intervals that are apart from each other. Let $I_1, I_2, \ldots, I_m$ $(m \geq 1)$ be these contiguous intervals. Then, we can divide all the flag jobs $\mathcal{F}$ into m groups, such that the span of each job group $\mathcal{H}_i$ is I_i, i.e., $\bigcup_{J\in\mathcal{H}_i} \Big[d(J), d(J)+p(J)\Big) = \Big[\min_{J\in\mathcal{H}_i} d(J), \max_{J\in\mathcal{H}_i} \Big(d(J)+p(J)\Big)\Big) = I_i$. The span of all the flag jobs can be written as $\sum_{i=1}^{m} \mathrm{len}(I_i)$.

By the argument for the Batch+ scheduler in the proof of Theorem 3.5, the active intervals of all the jobs started in the iteration defined by flag job J must lie inside the time interval $\Big[d(J), d(J)+(\alpha+1)\cdot p(J)\Big)$. Thus, the span of all the jobs $\mathcal{J}$ is bounded by $\mathrm{len}\Big(\bigcup_{J\in\mathcal{F}} \Big[d(J), d(J)+(\alpha+1)\cdot p(J)\Big)\Big)$. For each flag job $J \in \mathcal{H}_i$, we have $d(J) \geq \min_{J\in\mathcal{H}_i} d(J) = I_i^-$ and

$$\begin{aligned} & d(J)+(\alpha+1)\cdot p(J) \\ = \;& (\alpha+1)\cdot\Big(d(J)+p(J)\Big) - \alpha\cdot d(J) \\ \leq \;& (\alpha+1)\cdot \max_{J\in\mathcal{H}_i}\Big(d(J)+p(J)\Big) - \alpha\cdot \min_{J\in\mathcal{H}_i} d(J) \\ = \;& \min_{J\in\mathcal{H}_i} d(J) + (\alpha+1)\cdot\Big(\max_{J\in\mathcal{H}_i}\Big(d(J)+p(J)\Big) - \min_{J\in\mathcal{H}_i} d(J)\Big) \\ = \;& I_i^- + (\alpha+1)\cdot \mathrm{len}(I_i). \end{aligned}$$

Hence, $\Big[d(J), d(J)+(\alpha+1)\cdot p(J)\Big) \subseteq \Big[I_i^-, I_i^-+(\alpha+1)\cdot\mathrm{len}(I_i)\Big)$. This suggests that the span of all the jobs J is bounded by $\mathrm{len}\Big(\bigcup_{i=1}^{m}\Big[I_i^-, I_i^-+(\alpha+1)\cdot\mathrm{len}(I_i)\Big)\Big)$. Since $\bigcup_{i=1}^{m}\Big[I_i^-, I_i^-+(\alpha+1)\cdot\mathrm{len}(I_i)\Big)$ has a total length capped by $(\alpha+1)\cdot\sum_{i=1}^{m}\mathrm{len}(I_i)$, the lemma is proven. □

With Lemma 4.2, we only need to analyze the span for the flag jobs $\mathcal{F}$. Recall that $\mathcal{F}$ can be classified into the categories $\mathcal{F}_1$, $\mathcal{F}_2, \ldots, \mathcal{F}_m$, As argued in the proof of Theorem 3.5, each flag job designated by the Batch+ scheduler must arrive later than the latest possible completion time of the previous flag job. Thus, the active intervals of any two flag jobs in the same category cannot overlap with each other by any scheduler. In the following, we study the CDB-to-optimal span ratio[2] for any set of jobs with the above property.

Lemma 4.3. *If a set of jobs* $\mathcal{J}$ *can be classified into a collection of categories* $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_m$ *such that*

(a) the max/min job processing length ratio of each category is bounded by α*,*

(b) the active intervals of any two jobs in the same category cannot overlap by any scheduler,

then the CDB-to-optimal span ratio of $\mathcal{J}$ *is bounded by* $3+\frac{1}{\alpha-1}$.

Proof: We prove it by an induction on the number of job categories. If $\mathcal{J}$ contains only one job category, then according to property (b), the active intervals of any two jobs in $\mathcal{J}$ cannot overlap by any scheduler. Thus, the span induced by any scheduler is exactly $\sum_{J\in\mathcal{J}} p(J)$. Therefore, the CDB-to-optimal span ratio of $\mathcal{J}$ is 1, which is bounded by $3+\frac{1}{\alpha-1}$.

[2]For simplicity of presentation, we shall refer to the ratio between the spans induced by the Classify-by-Duration Batch+ scheduler and an optimal scheduler as the CDB-to-optimal span ratio.

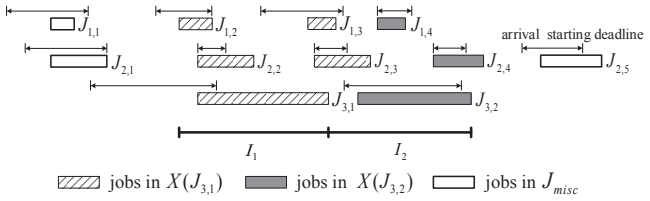

Figure 5: An optimal schedule induced by scheduler A

Suppose that the CDB-to-optimal span ratio is bounded by $3 + \frac{1}{\alpha-1}$ for all job sets containing no more than $(n-1)$ categories. We now show that for any job set $\mathcal{J}$ that contains n categories, its CDB-to-optimal span ratio is also bounded by $3 + \frac{1}{\alpha-1}$. Consider an optimal scheduler A on $\mathcal{J}$, i.e., $\text{span}_A(\mathcal{J}) = \text{span}_{min}(\mathcal{J})$. Let $I_A(J)$ denote the active interval of job $J \in \mathcal{J}$ induced by scheduler A.

We start by deriving a lower bound on the span induced by scheduler A. Assume that $\mathcal{J}$ can be classified into n categories $\mathcal{J}_1$, $\mathcal{J}_2$, ..., $\mathcal{J}_n$, where $\mathcal{J}_n$ includes jobs with the longest processing lengths. Let $J_{n,1}, J_{n,2}, \ldots, J_{n,q}$ be all the jobs in $\mathcal{J}_n$ sorted in an increasing order of their arrivals. The span induced by scheduler A for all the jobs $\mathcal{J}$ may consist of multiple contiguous time intervals. For each job $J_{n,j}$ $(1 \le j \le q)$, let $I_S(J_{n,j})$ denote the contiguous interval that $J_{n,j}$'s active interval $I_A(J_{n,j})$ falls in. Then, we define an interval I_j for $J_{n,j}$ as

$$I_j = \Big[\max\{I_S(J_{n,j})^-, I_A(J_{n,j-1})^+\}, I_A(J_{n,j})^+\Big),$$

i.e., I_j is left delimited by the starting point of $I_S(J_{n,j})$ and the ending point of $J_{n,j-1}$'s active interval, and it is right delimited by the ending point of $J_{n,j}$'s active interval. By property (b) of $\mathcal{J}_n$, we can infer that $I_A(J_{n,j}) \subseteq I_j$ and all the I_j's do not overlap with each other. For each I_j, define $X(J_{n,j})$ as the set of all the jobs $J \in \mathcal{J}$ satisfying $I_A(J) \subseteq I_j$ or $I_A(J)^- < I_j^+ < I_A(J)^+$, i.e., these jobs have their active intervals either fully contained in I_j or crossing the ending point of I_j. Obviously, $J_{n,j} \in X(J_{n,j})$. Let $\mathcal{J}_{overlap} = \bigcup_{j=1}^{q} X(J_{n,j})$ and $\mathcal{J}_{misc} = \mathcal{J} \backslash \mathcal{J}_{overlap}$. It is easy to see that the active interval of any job in $\mathcal{J}_{misc}$ does not overlap with any I_j, i.e., $\Big(\bigcup_{J\in\mathcal{J}_{misc}} I_A(J)\Big) \bigcap \Big(\bigcup_{j=1}^{q} I_j\Big) = \emptyset$. Figure 5 illustrates the above definitions with an example of jobs classified into three categories: $\mathcal{J}_1 = \{J_{1,1}, J_{1,2}, J_{1,3}, J_{1,4}\}$, $\mathcal{J}_2 = \{J_{2,1}, J_{2,2}, J_{2,3}, J_{2,4}, J_{2,5}\}$, $\mathcal{J}_3 = \{J_{3,1}, J_{3,2}\}$.

We can thus establish a lower bound on the span induced by the optimal scheduler A as

$$\begin{aligned} \text{span}_{min}(\mathcal{J}) &\ge \text{span}_A(\mathcal{J}_{misc}) + \sum_{j=1}^{q} \text{len}(I_j) \\ &\ge \text{span}_{min}(\mathcal{J}_{misc}) + \sum_{j=1}^{q} \text{len}(I_j), \qquad (1) \end{aligned}$$

where $\text{span}_{min}(\mathcal{J}_{misc})$ is the span induced by an optimal scheduler on $\mathcal{J}_{misc}$.

Next, we prove that the span induced by Classify-by-Duration Batch+ for $\mathcal{J}$ has the following upper bound:

$$\text{span}_{CDB}(\mathcal{J}) \le (3 + \frac{1}{\alpha-1}) \cdot \Big(\text{span}_{min}(\mathcal{J}_{misc}) + \sum_{j=1}^{q} \text{len}(I_j)\Big). \qquad (2)$$

Due to property (b), by applying Classify-by-Duration Batch+ on $\mathcal{J}$, each job in $\mathcal{J}$ is started at its starting deadline. Thus,

$$\begin{aligned} \text{span}_{CDB}(\mathcal{J}) &= \text{len}\Big(\bigcup_{J\in\mathcal{J}} \big[d(J), d(J)+p(J)\big)\Big) \\ &\le \text{len}\Big(\bigcup_{J\in\mathcal{J}_{misc}} \big[d(J), d(J)+p(J)\big)\Big) \\ &\quad + \text{len}\Big(\bigcup_{J\in\mathcal{J}_{overlap}} \big[d(J), d(J)+p(J)\big)\Big) \\ &= \text{span}_{CDB}(\mathcal{J}_{misc}) + \text{span}_{CDB}(\mathcal{J}_{overlap}). \end{aligned}$$

It is obvious that the jobs $\mathcal{J}_{misc}$ come from at most $(n-1)$ categories $\mathcal{J}_1$, $\mathcal{J}_2$, ..., $\mathcal{J}_{n-1}$. According to the induction hypothesis, the CDB-to-optimal span ratio of $\mathcal{J}_{misc}$ is bounded by $3 + \frac{1}{\alpha-1}$, i.e., $\text{span}_{CDB}(\mathcal{J}_{misc}) \le (3 + \frac{1}{\alpha-1}) \cdot \text{span}_{min}(\mathcal{J}_{misc})$. What remains is to show that $\text{span}_{CDB}(\mathcal{J}_{overlap}) \le (3 + \frac{1}{\alpha-1}) \cdot \sum_{j=1}^{q} \text{len}(I_j)$.

In each job set $X(J_{n,j})$ defined earlier, all the jobs other than $J_{n,j}$ fall in categories $\mathcal{J}_1$, $\mathcal{J}_2$, ..., $\mathcal{J}_{n-1}$. Recall that the active intervals of all the jobs in each category $\mathcal{J}_i$ cannot overlap by any scheduler. We can thus infer that $X(J_{n,j})$ contains at most one job $\widetilde{J}_i$ from each category $\mathcal{J}_i$ $(1 \le i \le n-1)$ which satisfies $I_A(J_{n,j})^+ < d(\widetilde{J}_i) + p(\widetilde{J}_i)$. Assume on the contrary that there are two jobs $\widetilde{J}_i$, $\widetilde{J}_i' \in X(J_{n,j}) \bigcap \mathcal{J}_i$ satisfying this condition. By the argument in the proof of Theorem 3.5, either $d(\widetilde{J}_i) + p(\widetilde{J}_i) \le a(\widetilde{J}_i')$ or $d(\widetilde{J}_i') + p(\widetilde{J}_i') \le a(\widetilde{J}_i)$ holds. This implies that one of these two jobs must arrive after time $I_A(J_{n,j})^+$. On the other hand, by the definition of $X(J_{n,j})$, these two jobs have their active intervals overlapping with I_j by scheduler A. So, they must both arrive before $I_j^+ = I_A(J_{n,j})^+$, which leads to a contradiction. Hence, there is at most one job $\widetilde{J}_i \in X(J_{n,j}) \bigcap \mathcal{J}_i$ satisfying $I_A(J_{n,j})^+ < d(\widetilde{J}_i) + p(\widetilde{J}_i)$. All the remaining jobs $J \in X(J_{n,j}) \bigcap \mathcal{J}_i \backslash \{\widetilde{J}_i\}$ must have their completion times $d(J) + p(J) \le I_A(J_{n,j})^+ = I_j^+$. By the definition of $X(J_{n,j})$, J's active interval $I_A(J)$ induced by scheduler A must be fully contained in I_j. Since $I_A(J)^- \le d(J)$, J's starting deadline $d(J)$ must also be in I_j. As a result, J's active interval $\big[d(J), d(J)+p(J)\big)$ induced by Classify-by-Duration Batch+ must be fully contained in I_j. Therefore,

$$\begin{aligned} &\text{len}\Big(\bigcup_{J\in X(J_{n,j})} \big[d(J), d(J)+p(J)\big)\Big) \\ &= \text{len}\Big(\bigcup_{i=1}^{n-1}\Big(\bigcup_{J\in X(J_{n,j})\cap\mathcal{J}_i} \big[d(J), d(J)+p(J)\big)\Big)\Big) \\ &\quad + \text{len}\Big(\big[d(J_{n,j}), d(J_{n,j})+p(J_{n,j})\big)\Big) \\ &\le \text{len}\Big(\bigcup_{i=1}^{n-1}\Big(\bigcup_{J\in X(J_{n,j})\cap\mathcal{J}_i\backslash\{\widetilde{J}_i\}} \big[d(J), d(J)+p(J)\big)\Big)\Big) \\ &\quad + \sum_{i=1}^{n-1} \text{len}\Big(\big[d(\widetilde{J}_i), d(\widetilde{J}_i)+p(\widetilde{J}_i)\big)\Big) + p(J_{n,j}) \\ &\le \text{len}(I_j) + \Big(\sum_{i=1}^{n-1} p(\widetilde{J}_i)\Big) + p(J_{n,j}). \end{aligned}$$

Since $\widetilde{J_i} \in \mathcal{J}_i$ and $J_{n,j} \in \mathcal{J}_n$, we have $p(\widetilde{J_i}) \leq \frac{1}{\alpha^{n-1-i}} \cdot p(J_{n,j})$. Note that $I_A(J_{n,j}) \subseteq I_j$, so $p(J_{n,j}) \leq \text{len}(I_j)$. It follows that $p(\widetilde{J_i}) \leq \frac{1}{\alpha^{n-1-i}} \cdot \text{len}(I_j)$. Thus,

$$\begin{aligned}
&\text{len}\Big(\bigcup_{J\in X(J_{n,j})}\big[d(J), d(J)+p(J)\big)\Big)\\
\leq\ & \text{len}(I_j) + \Big(\sum_{i=1}^{n-1}\frac{1}{\alpha^{n-1-i}}\cdot\text{len}(I_j)\Big) + \text{len}(I_j)\\
<\ & \text{len}(I_j) + \frac{1}{1-\frac{1}{\alpha}}\cdot\text{len}(I_j) + \text{len}(I_j)\\
=\ & (3+\frac{1}{\alpha-1})\cdot\text{len}(I_j).
\end{aligned}$$

Therefore,

$$\begin{aligned}
\text{span}_{CDB}(\mathcal{J}_{overlap}) &= \text{len}\Big(\bigcup_{J\in\mathcal{J}_{overlap}}\big[d(J), d(J)+p(J)\big)\Big)\\
&= \text{len}\Big(\bigcup_{j=1}^{q}\Big(\bigcup_{J\in X(J_{n,j})}\big[d(J), d(J)+p(J)\big)\Big)\Big)\\
&\leq \sum_{j=1}^{q}\text{len}\Big(\bigcup_{J\in X(J_{n,j})}\big[d(J), d(J)+p(J)\big)\Big)\\
&< (3+\frac{1}{\alpha-1})\cdot\sum_{i=1}^{q}\text{len}(I_j).
\end{aligned}$$

Hence, inequality (2) is proven. Based on (1) and (2), the CDB-to-optimal span ratio of the job set $\mathcal{J}$ with n categories is bounded by $3+\frac{1}{\alpha-1}$. By induction, the lemma is proven. □

By Lemma 4.3, the CDB-to-optimal span ratio for the set of flag jobs $\mathcal{F}$ is bounded by $3+\frac{1}{\alpha-1}$, i.e., $\frac{\text{span}_{CDB}(\mathcal{F})}{\text{span}_{min}(\mathcal{F})} \leq 3+\frac{1}{\alpha-1}$. It is obvious that the minimum possible span of any job set $\mathcal{J}$ is no less than that of its flag jobs $\mathcal{F}$ designated by Classify-by-Duration Batch+, i.e., $\text{span}_{min}(\mathcal{J}) \geq \text{span}_{min}(\mathcal{F})$. Thus, by Lemma 4.2, we have

$$\frac{\text{span}_{CDB}(\mathcal{J})}{\text{span}_{min}(\mathcal{J})} \leq \frac{(\alpha+1)\cdot\text{span}_{CDB}(\mathcal{F})}{\text{span}_{min}(\mathcal{F})} \leq 3\alpha+4+\frac{2}{\alpha-1}.$$

It is easy to derive that $(3\alpha+4+\frac{2}{\alpha-1})$ has a minimum value of $7+2\sqrt{6}\approx 11.9$ when α is set to $1+\sqrt{\frac{2}{3}}$.

Theorem 4.4. *The Classify-by-Duration Batch+ scheduler is $(3\alpha+4+\frac{2}{\alpha-1})$-competitive for Clairvoyant FJS, where α is the max/min job processing length ratio for each job category. When setting $\alpha = 1+\sqrt{\frac{2}{3}}$, this scheduler is $(7+2\sqrt{6})$-competitive.*

4.3 Profit Scheduler

Although the Classify-by-Duration Batch+ scheduler can achieve a constant competitive ratio, it schedules different categories of jobs independently, which may lose some opportunities to optimize the span. Moreover, Classify-by-Duration Batch+ does not make use of any runtime state in scheduling. In fact, with the knowledge of job processing length, we can derive how much the active interval of a new job can overlap with those of currently running jobs. This information would help decide whether starting a new job can make effective use of the existing span. Motivated by this observation, we propose a more competitive *Profit* scheduler that considers all the jobs as a whole.

The Profit scheduler determines the starting times of jobs in iterations. In each iteration, the Profit scheduler waits until a job J_f pending execution hits its starting deadline $d(J_f)$, i.e., J_f is the job with the earliest starting deadline among all pending jobs. Such a job J_f is designated as the *flag job* of this iteration. If several jobs share the same starting deadline, the one with the longest processing length is chosen as the flag job. The Profit scheduler starts the flag job J_f at time $d(J_f)$. For each job J that has arrived before time $d(J_f)$ but not yet started execution, if $p(J) \leq k\cdot p(J_f)$ (where $k>1$ is an algorithm parameter), the Profit scheduler also starts job J at time $d(J_f)$ in this iteration. This guarantees that at least $\frac{1}{k}$ of J's active interval overlaps with J_f's active interval in the span and such a job J is said to be *profitable* to job J_f. Apart from this, if a job J arrives during the active interval of the flag job J_f and satisfies $p(J) \leq k\cdot\Big(d(J_f)+p(J_f)-a(J)\Big)$, the Profit scheduler starts job J immediately at its arrival $a(J)$ in this iteration. Since the flag job J_f completes execution at time $d(J_f)+p(J_f)$, the above condition also ensures that at least $\frac{1}{k}$ of J's active interval overlaps with J_f's active interval in the span. Such a job J is also said to be *profitable* to job J_f. Meanwhile, among all the pending jobs not profitable to job J_f, the Profit scheduler watches for any job to hit its starting deadline. Once a job hits its starting deadline, a new iteration is initiated and the above job starting process is repeated. Note that by applying the Profit scheduler, the flag jobs of different iterations may be running concurrently. As a result, it is possible that a new incoming job J is profitable to several running flag jobs when it arrives. In this case, the scheduler starts job J immediately and we attribute J to any iteration of these flag jobs.

In the following text, we analyze the competitive ratio of the Profit scheduler. The analysis framework bears some similarity to that for Classify-by-Duration Batch+, but is more sophisticated than the latter. Let $\mathcal{F}$ be all the flag jobs designated by the Profit scheduler in the entire job set $\mathcal{J}$. The following lemma describes the relation between the spans induced by Profit for $\mathcal{J}$ and $\mathcal{F}$.

Lemma 4.5. *The span induced by the Profit scheduler for a set of jobs $\mathcal{J}$ is bounded by k times the span of all the flag jobs $\mathcal{F} \subseteq \mathcal{J}$ in the schedule.*

Proof: First, we show that each job J started in an iteration with a flag job J_f must have its active interval fall inside the time interval $\Big[d(J_f), d(J_f)+k\cdot p(J_f)\Big)$. If J arrives before time $d(J_f)$, then $p(J) \leq k\cdot p(J_f)$ and J is started at time $d(J_f)$. Then, the completion time of J satisfies $d(J_f)+p(J) \leq d(J_f)+k\cdot p(J_f)$. If J arrives during the active interval of the flag job J_f, then $p(J) \leq k\cdot\Big(d(J_f)+p(J_f)-a(J)\Big)$ and J is started at its arrival time $a(J)$. Hence, the completion time of job J satisfies

$$\begin{aligned}
a(J)+p(J) &\leq a(J)+k\cdot\Big(d(J_f)+p(J_f)-a(J)\Big)\\
&= a(J)+k\cdot p(J_f)-k\cdot(a(J)-d(J_f))\\
&< a(J)+k\cdot p(J_f)-(a(J)-d(J_f))\\
&= d(J_f)+k\cdot p(J_f).
\end{aligned}$$

Therefore, all the jobs started in an iteration with a flag job J_f must have their active intervals fall inside $\Big[d(J_f), d(J_f)+k\cdot p(J_f)\Big)$. As a result, the span of all the jobs $\mathcal{J}$ is bounded by $\text{len}\Big(\bigcup_{J\in\mathcal{F}}\Big[d(J), d(J)+k\cdot p(J)\Big)\Big)$.

By applying the Profit scheduler, the active interval of each flag job J_f is $\Big[d(J_f), d(J_f)+p(J_f)\Big)$. Thus, the span of all the flag jobs designated by the Profit scheduler is $\bigcup_{J\in\mathcal{F}}\Big[d(J), d(J)+p(J)\Big)$. Such a span may consist of multiple contiguous time intervals which are apart from each other. Let $I_1, I_2, \ldots, I_m$ ($m \geq 1$) be these contiguous intervals. Then, the span of all the flag jobs can be written as $\sum_{i=1}^{m}\text{len}(I_i)$. We can divide all the flag jobs $\mathcal{F}$ into m groups, such that the span of each job group $\mathcal{H}_i$ is I_i, i.e., $\bigcup_{J\in\mathcal{H}_i}\Big[d(J), d(J)+p(J)\Big) = \Big[\min_{J\in\mathcal{H}_i} d(J), \max_{J\in\mathcal{H}_i}\Big(d(J)+p(J)\Big)\Big) = I_i$. For each flag job $J\in\mathcal{H}_i$, we have $d(J) \geq \min_{J\in\mathcal{H}_i} d(J) = I_i^-$ and

$$\begin{aligned} & d(J)+k\cdot p(J) \\ = \;& k\cdot\Big(d(J)+p(J)\Big)-(k-1)\cdot d(J) \\ \leq \;& k\cdot\max_{J\in\mathcal{H}_i}\Big(d(J)+p(J)\Big)-(k-1)\cdot\min_{J\in\mathcal{H}_i} d(J) \\ = \;& \min_{J\in\mathcal{H}_i} d(J)+k\cdot\Big(\max_{J\in\mathcal{H}_i}\Big(d(J)+p(J)\Big)-\min_{J\in\mathcal{H}_i} d(J)\Big) \\ = \;& I_i^- + k\cdot\text{len}(I_i). \end{aligned}$$

Hence, $\Big[d(J), d(J)+k\cdot p(J)\Big) \subseteq \Big[I_i^-, I_i^- + k\cdot\text{len}(I_i)\Big)$. This suggests that the span of all the jobs $\mathcal{J}$ is bounded by $\text{len}\Big(\bigcup_{i=1}^{m}\Big[I_i^-, I_i^- + k\cdot\text{len}(I_i)\Big)\Big)$. Since $\bigcup_{i=1}^{m}\Big[I_i^-, I_i^- + k\cdot\text{len}(I_i)\Big)$ has a total length capped by $k\cdot\sum_{i=1}^{m}\text{len}(I_i)$, the lemma is proven. □

With Lemma 4.5, we only need to analyze the span for the flag jobs $\mathcal{F}$. We start by proving a useful lemma for the flag jobs $\mathcal{F}$.

Lemma 4.6. *For any two flag jobs designated by the Profit scheduler, the one with an earlier starting deadline must be completed before the other one.*

Proof: Let J_1 and J_2 be the flag jobs designated by the Profit scheduler in two different iterations. Without loss of generality, suppose J_1 has an earlier starting deadline than J_2, i.e., $d(J_1) < d(J_2)$.[3] If $a(J_2) \geq d(J_1)+p(J_1)$, then $d(J_2)+p(J_2) > a(J_2) \geq d(J_1)+p(J_1)$. If $a(J_2) < d(J_1)+p(J_1)$, since J_2 is also a flag job and is started at its starting deadline $d(J_2) > d(J_1)$, J_2 must not be profitable to J_1. If J_2 arrives before time $d(J_1)$, we have $p(J_2) > k\cdot p(J_1)$ and thus $d(J_2)+p(J_2) > d(J_1)+k\cdot p(J_1) > d(J_1)+p(J_1)$. If J_2 arrives during J_1's active interval $\Big[d(J_1), d(J_1)+p(J_1)\Big)$, we have $p(J_2) > k\cdot\Big(d(J_1)+p(J_1)-a(J_2)\Big)$ and it follows that

$$\begin{aligned} d(J_2)+p(J_2) &> a(J_2)+k\cdot\Big(d(J_1)+p(J_1)-a(J_2)\Big) \\ &> a(J_2)+\Big(d(J_1)+p(J_1)-a(J_2)\Big) \\ &= d(J_1)+p(J_1). \end{aligned}$$

Therefore, in any case, J_1 is completed before J_2. □

To study the Profit-to-optimal span ratio,[4] we construct a directed graph $G(\mathcal{F}, E)$ for the set of flag jobs $\mathcal{F}$ as follows. For each flag job $J\in\mathcal{F}$, let $X(J)$ be the set of all the flag jobs $J'\in\mathcal{F}$ satisfying $a(J') < d(J)+p(J)$ and $d(J) < d(J')$, i.e., J' arrives before J completes and J' is started after J. Thus, J' must not be profitable to J. If $X(J)\neq\emptyset$, then among all the jobs in $X(J)$, we choose the job $\widetilde{J}$ with the earliest starting deadline and create a directed edge from $\widetilde{J}$ to J in the graph $G(\mathcal{F}, E)$. Figure 6 shows an example of graph construction. It can be shown that the graph constructed in this way is a collection of rooted trees.

Lemma 4.7. *The directed graph $G(\mathcal{F}, E)$ is a collection of rooted trees.*

Proof: By definition, for each job $J\in\mathcal{F}$ with $X(J)\neq\emptyset$, there is only one job $J'\in X(J)$ which has an edge pointing to J. Moreover, no cycle exists in $G(\mathcal{F}, E)$. Otherwise, suppose a set of edges (J_1, J_2), (J_2, J_3), …, (J_{m-1}, J_m), $(J_m, J_1)\in E$ form a cycle. According to the definition of an edge $(J', J)\in E$, we have $d(J) < d(J')$. Consequently, $d(J_1) < d(J_m) < d(J_{m-1}) < \cdots < d(J_3) < d(J_2) < d(J_1)$, which leads to a contradiction. Hence, the lemma is proven. □

Based on Lemma 4.7, for each edge $(J', J)\in E$, we refer to job J' as the *parent* of job J, and refer to J as a *child* of J'. By definition, if job J' is the parent of job J, J' is the job with the earliest starting deadline in $X(J)$. If $X(J)=\emptyset$ for a job J, we refer to J as a *root job*. If a job J does not have any child, we call it a *leaf job*. We next show that to study the Profit-to-optimal span ratio for all the flag jobs $\mathcal{F}$, it is equivalent to study the Profit-to-optimal span ratio for any rooted tree in the graph $G(\mathcal{F}, E)$.

Lemma 4.8. *Given a job $J\in\mathcal{F}$, for each job $J'\in X(J)$, there exists a path from J' to J in the graph $G(\mathcal{F}, E)$.*

Proof: Suppose job J is in a rooted tree T of $G(\mathcal{F}, E)$. Let $J_1, J_2, \ldots, J_q, J$ be the sequence of nodes along the path from the root job of tree T to J, i.e., J_1 is the root job of T and is the parent of J_2, J_2 is the parent of $J_3, \ldots$, and J_q is the parent of J. By the definition of edges, we have $d(J) < d(J_q) < d(J_{q-1}) < \cdots < d(J_1)$. We prove that $X(J)\subseteq\{J_1, J_2, \ldots, J_q\}$ by contradiction.

Assume on the contrary that there exists a job $J'\in X(J)$ such that $J'\notin\{J_1, J_2, \ldots, J_q\}$. Since J_q is the parent of J, J_q has the earliest starting deadline among all the jobs in $X(J)$. Thus, $d(J') > d(J_q)$.

We first consider the case that $d(J_i) < d(J') < d(J_{i-1})$ for some i. Since $J'\in X(J)$, we have $a(J') < d(J)+p(J)$. It follows from $d(J) < d(J_i)$ and Lemma 4.6 that $d(J)+p(J) < d(J_i)+p(J_i)$. As a result, $a(J') < d(J_i)+p(J_i)$ and thus $J'\in X(J_i)$. The existence of the edge (J_{i-1}, J_i) suggests that job J_{i-1} has the earliest starting deadline among all the jobs in $X(J_i)$. Therefore, $d(J_{i-1}) < d(J')$, which leads to a contradiction.

Next, we consider the case that $d(J_1) < d(J')$. Similarly, since $a(J') < d(J)+p(J) < d(J_1)+p(J_1)$, we have $J'\in X(J_1)$, but this contradicts the fact that $X(J_1)=\emptyset$ since J_1 is a root job.

[3] Note that $d(J_1)\neq d(J_2)$. Otherwise, at their (same) starting deadlines, the job with a longer processing length would be chosen as the flag job by the Profit scheduler. Then, the other job is profitable to the flag job and will be started in the same iteration so that it would not become a flag job itself.

[4] For simplicity of presentation, we shall refer to the ratio between the spans induced by the Profit scheduler and an optimal scheduler as the Profit-to-optimal span ratio.

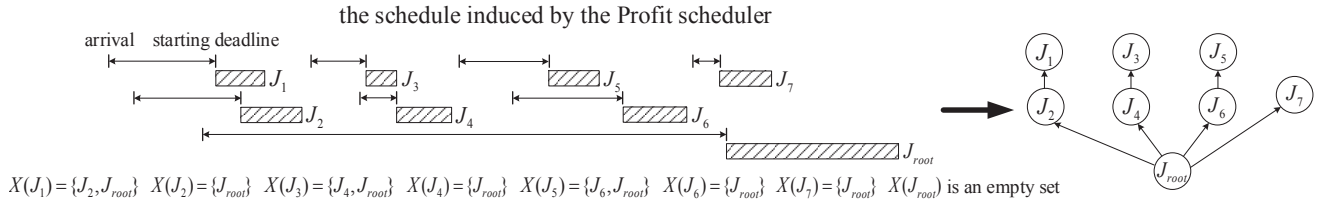

Figure 6: Constructing a graph for flag jobs

Thus, it is proven that $X(J) \subseteq \{J_1, J_2, \ldots, J_q\}$, which implies that there exists a path from each job $J' \in X(J)$ to job J. □

Lemma 4.9. *If there is no path between two jobs J and J' in the graph $G(\mathcal{F}, E)$, the active intervals of J and J' can never overlap with each other by any scheduler.*

Proof: Without loss of generality, suppose $d(J) < d(J')$. If $a(J') < d(J) + p(J)$, then $J' \in X(J)$. According to Lemma 4.8, there exists a path from J' to J in the graph $G(\mathcal{F}, E)$, which leads to a contradiction. Therefore, we have $a(J') \geq d(J) + p(J)$, i.e., job J' arrives no earlier than the latest possible completion time of job J. Thus, their active intervals cannot overlap with each other by any scheduler. □

Lemma 4.10. *The Profit-to-optimal span ratio for the flag jobs $\mathcal{F}$ is bounded by a constant C if the Profit-to-optimal span ratio for the flag jobs in any rooted tree of the graph $G(\mathcal{F}, E)$ is bounded by C.*

Proof: According to Lemma 4.7, we can decompose the directed graph $G(\mathcal{F}, E)$ into a collection of rooted trees $T_1(\mathcal{F}_1, E_1)$, $T_2(\mathcal{F}_2, E_2), \ldots, T_m(\mathcal{F}_m, E_m)$, where m is an integer and $\mathcal{F}_i$ is the set of flag jobs in tree T_i. For any two jobs $J \in \mathcal{F}_i$ and $J' \in \mathcal{F}_j$ $(i \neq j)$, there is no path between them in $G(\mathcal{F}, E)$. By Lemma 4.9, the active intervals of J and J' cannot overlap with each other by any scheduler. This suggests that an optimal schedule for the job set $\mathcal{F}_i \cup \mathcal{F}_j$ can be obtained by simply combining the optimal schedules for $\mathcal{F}_i$ and $\mathcal{F}_j$, and thus $\text{span}_{min}(\mathcal{F}_i \cup \mathcal{F}_j) = \text{span}_{min}(\mathcal{F}_i) + \text{span}_{min}(\mathcal{F}_j)$. Consequently, $\text{span}_{min}(\mathcal{F}) = \sum_{i=1}^{m} \text{span}_{min}(\mathcal{F}_i)$.

Note that by applying the Profit scheduler, all the flag jobs are started at their starting deadlines. Thus,

$$\begin{aligned}
\text{span}_{Profit}(\mathcal{F}) &= \text{len}\Big(\bigcup_{J\in\mathcal{F}} \big[d(J), d(J)+p(J)\big)\Big) \\
&\leq \sum_{i=1}^{m} \text{len}\Big(\bigcup_{J\in\mathcal{F}_i} \big[d(J), d(J)+p(J)\big)\Big) \\
&= \sum_{i=1}^{m} \text{span}_{Profit}(\mathcal{F}_i).
\end{aligned}$$

If the Profit-to-optimal span ratio for any $\mathcal{F}_i$ is bounded by C, i.e., $\text{span}_{Profit}(\mathcal{F}_i) \leq C \cdot \text{span}_{min}(\mathcal{F}_i)$, it follows that $\text{span}_{Profit}(\mathcal{F}) \leq \sum_{i=1}^{m} \big(C \cdot \text{span}_{min}(\mathcal{F}_i)\big) = C \cdot \text{span}_{min}(\mathcal{F})$. □

We now focus on the Profit-to-optimal span ratio for a rooted tree of flag jobs. In the following, we prove an upper bound of $2 + \frac{1}{k} + \frac{1}{k-1}$ on the Profit-to-optimal span ratio for any rooted tree by an induction on the *height* of the rooted tree — the number of edges along the longest path between the root job and the leaf jobs in the rooted tree.

First, consider a rooted tree with a height of 0, i.e., there is only one job J in the tree. Then, the span induced by any scheduler is exactly $p(J)$. Thus, the Profit-to-optimal span ratio for such a tree is 1, which is bounded by $2 + \frac{1}{k} + \frac{1}{k-1}$.

Suppose that the Profit-to-optimal span ratio is bounded by $2 + \frac{1}{k} + \frac{1}{k-1}$ for any rooted tree with a height no more than $(n-1)$. According to Lemma 4.10, this upper bound of the span ratio also holds for a directed graph consisting of a collection of rooted trees, each with a height no more than $(n-1)$. We now show that this upper bound also holds for any rooted tree T with a height of n. Let $\mathcal{J}_T$ be the set of flag jobs in T. Consider an optimal scheduler A on $\mathcal{J}_T$, i.e., $\text{span}_A(\mathcal{J}_T) = \text{span}_{min}(\mathcal{J}_T)$. Let $I_A(J)$ denote the active interval of a job $J \in \mathcal{J}_T$ induced by scheduler A.

We start by determining a lower bound on the span induced by scheduler A. The span induced by scheduler A for the flag jobs $\mathcal{J}_T$ may consist of multiple contiguous time intervals. Let J_{root} be the root job of T and let I_S denote the contiguous interval that J_{root}'s active interval $I_A(J_{root})$ falls in. As illustrated in Figure 7, we define an interval $I_{middle} = \big[I_S^-, I_A(J_{root})^+\big)$, i.e., I_{middle} is left delimited by the starting point of I_S and it is right delimited by the ending point of J_{root}'s active interval. It is obvious that $I_A(J_{root}) \subseteq I_{middle}$ and thus $p(J_{root}) \leq \text{len}(I_{middle})$. Then, we define $\mathcal{J}_{middle}$ as the set of all the jobs $J \in \mathcal{J}_T$ satisfying $I_A(J) \subseteq I_{middle}$ or $I_A(J)^- < I_{middle}^+ < I_A(J)^+$, i.e., these jobs have their active intervals either fully contained in I_{middle} or crossing the ending point of I_{middle}. Obviously, $J_{root} \in \mathcal{J}_{middle}$. Let $\mathcal{J}_{left}$ be the set of all the jobs $J \in \mathcal{J}_T$ satisfying $I_A(J)^+ < I_{middle}^-$. Let $\mathcal{J}_{right}$ be the set of all the jobs other than $\mathcal{J}_{left}$ and $\mathcal{J}_{middle}$, i.e., $\mathcal{J}_{right} = \mathcal{J}_T \backslash (\mathcal{J}_{left} \cup \mathcal{J}_{middle})$. Apparently, each job $J \in \mathcal{J}_{right}$ satisfies $I_A(J)^- \geq I_{middle}^+ = I_A(J_{root})^+$. We can therefore establish a lower bound on the span induced by the optimal scheduler A as

$$\begin{aligned}
&\text{span}_{min}(\mathcal{J}_T) \\
&\quad\geq \text{span}_A(\mathcal{J}_{left}) + \text{len}(I_{middle}) + \text{span}_A(\mathcal{J}_{right}) \\
&\quad\geq \text{span}_{min}(\mathcal{J}_{left}) + \text{len}(I_{middle}) + \text{span}_{min}(\mathcal{J}_{right}), \quad (3)
\end{aligned}$$

where $\text{span}_{min}(\mathcal{J}_{left})$ and $\text{span}_{min}(\mathcal{J}_{right})$ are the spans induced by the optimal schedulers on $\mathcal{J}_{left}$ and $\mathcal{J}_{right}$ respectively.

Next, we prove that the span induced by the Profit scheduler for $\mathcal{J}_T$, i.e., $\text{span}_{Profit}(\mathcal{J}_T)$, is bounded by $(2+\frac{1}{k}+\frac{1}{k-1}) \cdot \big(\text{span}_{min}(\mathcal{J}_{left}) + \text{len}(I_{middle}) + \text{span}_{min}(\mathcal{J}_{right})\big)$.

Note that all the jobs in $\mathcal{J}_T$ are flag jobs. Thus, by applying the Profit scheduler, each job in $\mathcal{J}_T$ is started at its starting deadline. It follows that

$$\text{span}_{Profit}(\mathcal{J}_T) = \text{len}\Big(\bigcup_{J\in\mathcal{J}_T} \big[d(J), d(J)+p(J)\big)\Big)$$

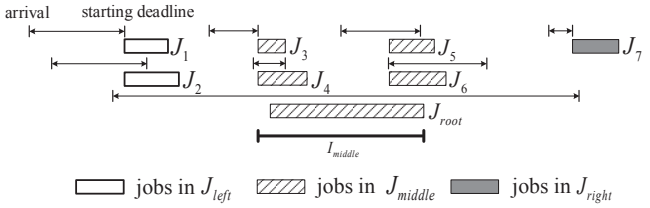

Figure 7: An optimal schedule on a rooted tree

$$\begin{aligned}
&\leq \operatorname{len}\Big(\bigcup_{J\in\mathcal{J}_{left}}\big[d(J),d(J)+p(J)\big)\Big)\\
&\quad+\operatorname{len}\Big(\bigcup_{J\in\mathcal{J}_{middle}}\big[d(J),d(J)+p(J)\big)\Big)\\
&\quad+\operatorname{len}\Big(\bigcup_{J\in\mathcal{J}_{right}}\big[d(J),d(J)+p(J)\big)\Big)\\
&= \operatorname{span}_{Profit}(\mathcal{J}_{left})+\operatorname{span}_{Profit}(\mathcal{J}_{middle})\\
&\quad+\operatorname{span}_{Profit}(\mathcal{J}_{right}).
\end{aligned}$$

It can be shown that by applying the graph construction technique for the job set $\mathcal{J}_{left}$, $\mathcal{J}_{left}$ must form a collection of rooted trees, each with a height no more than $(n-1)$. Let $G'(\mathcal{J}_{left}, E_{left})$ be the graph constructed for $\mathcal{J}_{left}$. For each edge $(J, J') \in E_{left}$, by definition, we have $J \in X(J')$. It follows from Lemma 4.8 that there exists a path from J to J' in the rooted tree T of jobs $\mathcal{J}_T$. Thus, every path in G' remains and cannot become shorter in T. Note that $J_{root} \notin \mathcal{J}_{left}$ is a root job that has a path to every node in T. Therefore, the longest path in T must be longer than that in G'. This implies that each rooted tree in G' has height at most $(n-1)$. According to the induction hypothesis and Lemma 4.10, the Profit-to-optimal span ratio of $\mathcal{J}_{left}$ is bounded by $2+\frac{1}{k}+\frac{1}{k-1}$, i.e., $\operatorname{span}_{Profit}(\mathcal{J}_{left}) \leq (2+\frac{1}{k}+\frac{1}{k-1})\cdot\operatorname{span}_{min}(\mathcal{J}_{left})$. Similarly, we have $\operatorname{span}_{Profit}(\mathcal{J}_{right}) \leq (2+\frac{1}{k}+\frac{1}{k-1})\cdot\operatorname{span}_{min}(\mathcal{J}_{right})$. What remains is to show that $\operatorname{span}_{Profit}(\mathcal{J}_{middle}) \leq (2+\frac{1}{k}+\frac{1}{k-1})\cdot\operatorname{len}(I_{middle})$.

Recall that each job in $\mathcal{J}_{middle}$ is started at its starting deadline by the Profit scheduler. This suggests that the starting time of each job $J \in \mathcal{J}_{middle}$ determined by the Profit scheduler must be no earlier than $I_A(J)^-$. Let $J_1, J_2, \ldots, J_m$ be all the jobs in $\mathcal{J}_{middle}\backslash\{J_{root}\}$ sorted in an increasing order of their starting deadlines. Since J_{root} is the root job, by the definition of edges, J_{root} must have a later starting deadline than all the other jobs in $\mathcal{J}_{middle}$. As a result, we have $d(J_1) < d(J_2) < \cdots < d(J_m) < d(J_{root})$. Let J_i be the highest-indexed job satisfying $d(J_i) \leq I_A(J_{root})^+$, if there exists at least one such job. For notational convenience, if such a job does not exist (i.e., $I_A(J_{root})^+ < d(J_1)$), we define $i = 0$.

We first show that the span induced by the Profit scheduler for jobs $\{J_1, J_2, \ldots, J_i\}$ is at most $\operatorname{len}(I_{middle})+\frac{1}{k}\cdot p(J_{root})$. According to Lemma 4.6, the jobs $J_1, J_2, \ldots, J_i$ must be completed no later than time $d(J_i)+p(J_i)$. If $d(J_i)+p(J_i) \leq I_A(J_{root})^+$, all these jobs must have their active intervals fully contained in I_{middle}. Consequently, the span induced by the Profit scheduler for these jobs is at most $\operatorname{len}(I_{middle})$. If $d(J_i)+p(J_i) > I_A(J_{root})^+$, the span induced by the Profit scheduler for jobs $\{J_1, J_2, \ldots, J_i\}$ is at most

$$\begin{aligned}
&\operatorname{len}\Big(\big[I^-_{middle}, d(J_i)+p(J_i)\big)\Big)\\
&= \operatorname{len}(I_{middle})+\operatorname{len}\Big(\big[I_A(J_{root})^+, d(J_i)+p(J_i)\big)\Big). \quad (4)
\end{aligned}$$

Since $I_A(J_{root})^+ \in \big[d(J_i), d(J_i)+p(J_i)\big)$, job J_{root} must arrive before job J_i completes in the Profit schedule. This implies that job J_{root} is not profitable to J_i. If J_{root} arrives earlier than time $d(J_i)$, we have $p(J_{root}) > k\cdot p(J_i)$. It follows from $d(J_i) \leq I_A(J_{root})^+$ that $p(J_{root}) > k\cdot\big(d(J_i)+p(J_i)-I_A(J_{root})^+\big) = k\cdot\operatorname{len}\Big(\big[I_A(J_{root})^+, d(J_i)+p(J_i)\big)\Big)$. If J_{root} arrives during J_i's active interval $\big[d(J_i), d(J_i)+p(J_i)\big)$, we have $p(J_{root}) > k\cdot\big(d(J_i)+p(J_i)-a(J_{root})\big)$. Note that $a(J_{root}) \leq I_A(J_{root})^- < I_A(J_{root})^+$. Hence, we have $p(J_{root}) > k\cdot\big(d(J_i)+p(J_i)-I_A(J_{root})^+\big) = k\cdot\operatorname{len}\Big(\big[I_A(J_{root})^+, d(J_i)+p(J_i)\big)\Big)$. Therefore, by inequality (4), the span for the jobs $\{J_1, J_2, \ldots, J_i\}$ is always bounded by $\operatorname{len}(I_{middle})+\frac{1}{k}\cdot p(J_{root})$.

We next show that if $i < m$, $p(J_{j+1}) > k\cdot p(J_j)$ holds for each $i+1 \leq j \leq m-1$, and $p(J_{root}) > k\cdot p(J_m)$. By the definition of $\mathcal{J}_{middle}$, we have $I_A(J_{j+1})^- < I_A(J_{root})^+$. It follows that $a(J_{j+1}) \leq I_A(J_{j+1})^- < I_A(J_{root})^+ < d(J_{i+1}) \leq d(J_j)$. Thus, when job J_j is started at its starting deadline $d(J_j)$, J_{j+1} must have arrived and is not profitable to J_j, which suggests that $p(J_{j+1}) > k\cdot p(J_j)$. Similarly, since $a(J_{root}) \leq I_A(J_{root})^- < I_A(J_{root})^+ < d(J_{i+1}) \leq d(J_m)$, J_{root} is not profitable to J_m. Hence, $p(J_{root}) > k\cdot p(J_m)$. Together, these relations suggest that $p(J_j) < \frac{1}{k^{m+1-j}}\cdot p(J_{root})$ for each $i+1 \leq j \leq m$.

Therefore, the span induced by the Profit scheduler for $\mathcal{J}_{middle}$ satisfies

$$\begin{aligned}
&\operatorname{span}_{Profit}(\mathcal{J}_{middle})\\
&\leq \operatorname{len}\Big(\bigcup_{j=1}^{i}\big[d(J_j), d(J_j)+p(J_j)\big)\Big)\\
&\quad+\operatorname{len}\Big(\bigcup_{j=i+1}^{m}\big[d(J_j), d(J_j)+p(J_j)\big)\Big)\\
&\quad+\operatorname{len}\Big(\big[d(J_{root}), d(J_{root})+p(J_{root})\big)\Big)\\
&\leq \operatorname{len}(I_{middle})+\frac{1}{k}\cdot p(J_{root})+\sum_{j=i+1}^{m}p(J_j)+p(J_{root})\\
&< \operatorname{len}(I_{middle})+(1+\frac{1}{k})\cdot p(J_{root})+\Big(\sum_{h=1}^{\infty}\frac{1}{k^h}\Big)\cdot p(J_{root})\\
&= \operatorname{len}(I_{middle})+(1+\frac{1}{k}+\frac{1}{k-1})\cdot p(J_{root})\\
&\leq (2+\frac{1}{k}+\frac{1}{k-1})\cdot\operatorname{len}(I_{middle}).
\end{aligned}$$

Now, according to inequality (3), the Profit-to-optimal span ratio for the flag jobs $\mathcal{J}_T$ in a rooted tree T with a height of n is bounded by $2+\frac{1}{k}+\frac{1}{k-1}$. By induction, we can conclude that the Profit-to-optimal span ratio for any rooted tree of flag jobs is bounded by $2+\frac{1}{k}+\frac{1}{k-1}$.

By Lemma 4.10, the Profit-to-optimal span ratio for all the flag jobs $\mathcal{F}$ is bounded by $2+\frac{1}{k}+\frac{1}{k-1}$. It is apparent that the minimum possible span of any job set $\mathcal{J}$ is no less than that of its flag jobs $\mathcal{F}$ designated by the Profit scheduler, i.e., $\operatorname{span}_{min}(\mathcal{J}) \geq$

$\text{span}_{min}(\mathcal{F})$. Thus, by Lemma 4.5, we have

$$\frac{\text{span}_{Profit}(\mathcal{J})}{\text{span}_{min}(\mathcal{J})} \leq \frac{k \cdot \text{span}_{Profit}(\mathcal{F})}{\text{span}_{min}(\mathcal{F})} \leq 2k + 2 + \frac{1}{k-1}.$$

It is easy to derive that $2k + 2 + \frac{1}{k-1}$ has a minimum value of $4 + 2\sqrt{2} \approx 6.9$ when k is set to $1 + \frac{\sqrt{2}}{2}$.

Theorem 4.11. *The Profit scheduler is $(2k + 2 + \frac{1}{k-1})$-competitive for Clairvoyant FJS. When setting $k = 1 + \frac{\sqrt{2}}{2}$, this scheduler is $(2\sqrt{2} + 4)$-competitive.*

5 CONCLUDING REMARKS

We briefly outline how our results can be used to extend the MinUsageTime Dynamic Bin Packing (DBP) problem. The MinUsageTime DBP problem was defined to model server acquisition and job scheduling in cloud-based systems [15, 16, 19, 23]. In this problem, a set of items (jobs) are to be packed into bins (cloud servers). Each item has an arrival time, a departure time and a size. The item has to be placed in a bin from its arrival to its departure. The size of the item models the resource demand of the job. The total size of the items placed in one bin cannot exceed the bin capacity (server capacity) at any time. The target is to minimize the total usage time of all the bins used for packing. Competitiveness analysis has been carried out for various packing algorithms in the non-clairvoyant and clairvoyant settings [15, 16, 19, 20, 23]. It has been shown that the total bin usage time is bounded by adding the span of all the items and the accumulated time-space demand of all the items. So far, the jobs modeled by MinUsageTime DBP are restricted to "rigid" jobs that must be started immediately at their arrivals. In this case, the span of all the jobs is fixed and independent of the packing algorithm.

To generalize MinUsageTime DBP and model flexible jobs that have laxity in starting, we can apply a scheduling algorithm to determine the starting time of each job and apply a packing algorithm to decide where to place (run) the job when it starts. In this way, the span of all the jobs is affected by the scheduling algorithm. In the non-clairvoyant setting, it is known that a First Fit packing algorithm can achieve a near-optimal $O(\mu)$ competitive ratio for MinUsageTime DBP [20, 23]. Since our Batch+ scheduler is $O(\mu)$-competitive for minimizing the span, integrating Batch+ scheduling with First Fit packing can achieve an $O(\mu)$ competitive ratio for generalized MinUsageTime DBP. In the clairvoyant setting, applying a classify-by-duration strategy to First Fit packing can achieve an $O(\log \mu)$ competitive ratio for MinUsageTime DBP [19]. Since our Profit scheduler is $O(1)$-competitive for minimizing the span, combining Profit scheduling with classify-by-duration First Fit packing can achieve an $O(\log \mu)$ competitive ratio for generalized MinUsageTime DBP.

It was recently brought to our attention a concurrent work by Koehler *et al.* [12] which studied a busy-time scheduling problem on a bounded number of machines. An online case of their problem when the machine capacity is unbounded is equivalent to our Clairvoyant FJS. They independently established the same lower bound of $\frac{\sqrt{5}+1}{2}$ on the competitive ratio of any online scheduler, and proposed a 5-competitive *Doubler* scheduler.

6 ACKNOWLEDGMENTS

This work is supported by Singapore Ministry of Education Academic Research Fund Tier 2 under Grant MOE2013-T2-2-067, and Academic Research Fund Tier 1 under Grant 2013-T1-002-123.

REFERENCES

[1] Amazon EC2 Pricing. http://aws.amazon.com/ec2/pricing/. (2017).
[2] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. 2001. A unified approach to approximating resource allocation and scheduling. *J. ACM* 48, 5 (2001), 735–744.
[3] Amotz Bar-Noy, Sudipto Guha, Joseph Naor, and Baruch Schieber. 2001. Approximating the throughput of multiple machines in real-time scheduling. *SIAM J. Comput.* 31, 2 (2001), 331–352.
[4] Luiz André Barroso and Urs Hölzle. 2007. The Case for Energy-Proportional Computing. *Computer* 40, 12 (2007), 33–37.
[5] Sanjoy Baruah, Gilad Koren, Decao Mao, Bhubaneswar Mishra, Arvind Raghunathan, Louis Rosier, Dennis Shasha, and Fuxing Wang. 1991. On the competitiveness of on-line real-time task scheduling. In *Proceedings of the 12th IEEE Real-Time Systems Symposium (RTSS)*. 106–115.
[6] Allan Borodin and Ran El-Yaniv. 1998. *Online computation and competitive analysis*. Vol. 53. Cambridge University Press.
[7] Jessica Chang, Samir Khuller, and Koyel Mukherjee. 2014. LP rounding and combinatorial algorithms for minimizing active and busy time. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 118–127.
[8] Julia Chuzhoy, Rafail Ostrovsky, and Yuval Rabani. 2006. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research* 31, 4 (2006), 730–738.
[9] Ulrich Faigle and Willem M. Nawijn. 1995. Note on scheduling intervals on-line. *Discrete Applied Mathematics* 58, 1 (1995), 13–17.
[10] Michele Flammini, Gianpiero Monaco, Luca Moscardelli, Hadas Shachnai, Mordechai Shalom, Tami Tamir, and Shmuel Zaks. 2010. Minimizing total busy time in parallel scheduling with application to optical networks. *Theoretical Computer Science* 411, 40-42 (2010), 3553–3562.
[11] Rohit Khandekar, Baruch Schieber, Hadas Shachnai, and Tami Tamir. 2015. Real-time scheduling to minimize machine busy times. *Journal of Scheduling* 18, 6 (2015), 561–573.
[12] Frederic Koehler and Samir Khuller. 2017. Busy-Time Scheduling on a Bounded Number of Machines. In *Proceedings of the 15th Algorithms and Data Structures Symposium (WADS)*, to appear.
[13] Antoon W.J. Kolen, Jan Karel Lenstra, Christos H. Papadimitriou, and Frits C.R. Spieksma. 2007. Interval scheduling: A survey. *Naval Research Logistics* 54, 5 (2007), 530–543.
[14] Mikhail Y. Kovalyov, C.T. Ng, and T.C. Edwin Cheng. 2007. Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European Journal of Operational Research* 178, 2 (2007), 331–342.
[15] Yusen Li, Xueyan Tang, and Wentong Cai. 2014. On dynamic bin packing for resource allocation in the cloud. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 2–11.
[16] Yusen Li, Xueyan Tang, and Wentong Cai. 2016. Dynamic Bin Packing for On-Demand Cloud Resource Allocation. *IEEE Transactions on Parallel and Distributed Systems* 27, 1 (2016), 157–170.
[17] Richard J. Lipton and Andrew Tomkins. 1994. Online Interval Scheduling. In *Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 302–311.
[18] George B. Mertzios, Mordechai Shalom, Ariella Voloshin, Prudence W.H. Wong, and Shmuel Zaks. 2015. Optimizing busy time on parallel machines. *Theoretical Computer Science* 562 (2015), 524–541.
[19] Runtian Ren and Xueyan Tang. 2016. Clairvoyant Dynamic Bin Packing for Job Scheduling with Minimum Server Usage Time. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 227–237.
[20] Runtian Ren, Xueyan Tang, Yusen Li, and Wentong Cai. 2016. Competitiveness of Dynamic Bin Packing for Online Cloud Server Allocation. *IEEE/ACM Transactions on Networking* (2016).
[21] Steven S. Seiden. 1998. Randomized online interval scheduling. *Operations Research Letters* 22, 4 (1998), 171–177.
[22] Mordechai Shalom, Ariella Voloshin, Prudence W.H. Wong, Fencol C.C. Yung, and Shmuel Zaks. 2014. Online optimization of busy time on parallel machines. *Theoretical Computer Science* 560 (2014), 190–206.
[23] Xueyan Tang, Yusen Li, Runtian Ren, and Wentong Cai. 2016. On First Fit Bin Packing for Online Cloud Server Allocation. In *Proceedings of the 30th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 323–332.
[24] Gerhard J. Woeginger. 1994. On-line scheduling of jobs with fixed start and end times. *Theoretical Computer Science* 130, 1 (1994), 5–16.