Dispatching Equal-length Jobs to Parallel Machines to Maximize Throughput

David P. Bunde[1] and Michael H. Goldwasser[2]

[1] Dept. of Computer Science, Knox College
email: dbunde@knox.edu
[2] Dept. of Mathematics and Computer Science, Saint Louis University
email: goldwamh@slu.edu

Abstract. We consider online, nonpreemptive scheduling of equal-length jobs on parallel machines. Jobs have arbitrary release times and deadlines and a scheduler's goal is to maximize the number of completed jobs ($Pm \mid r_j, p_j = p \mid \sum 1 - U_j$). This problem has been previously studied under two distinct models. In the first, a scheduler must provide *immediate notification* to a released job as to whether it is accepted into the system. In a stricter model, a scheduler must provide an *immediate decision* for an accepted job, selecting both the time interval and machine on which it will run. We examine an intermediate model in which a scheduler *immediately dispatches* an accepted job to a machine, but without committing it to a specific time interval. We present a natural algorithm that is optimally competitive for $m = 2$. For the special case of unit-length jobs, it achieves competitive ratios for $m \geq 2$ that are strictly better than lower bounds for the immediate decision model.

1 Introduction

We consider a model in which a scheduler manages a pool of parallel machines. Job requests arrive in an online fashion, and the scheduler receives credit for each job that is completed by its deadline. We assume that jobs have equal length and that the system is nonpreemptive. We examine a series of increasingly restrictive conditions on the timing of a scheduler's decisions. Specifically, we consider the following submodels.

unrestricted: In this most flexible model, all requests are pooled by a scheduler. Decisions are made in real-time, with jobs dropped only when it is clear they will not be completed on time.

immediate notification: In this model, the scheduler must decide whether a job will be admitted to the system when it arrives. Once admitted, a job must be completed on time. However, the scheduler retains flexibility by centrally pooling admitted jobs until they are executed.

immediate dispatch: In this model, a central scheduler must immediately assign an admitted job to a particular machine, but each machine retains autonomy in determining the order in which to execute the jobs assigned to it, provided they are completed on time.

immediate decision: In this model, a central scheduler must fully commit an admitted job to a particular machine and to a particular time interval for execution on that machine.

The problem has been previously studied in the unrestricted, immediate notification, and immediate decision models. Immediate dispatching is motivated by multiprocessor settings where incoming requests to a server farm or computer cluster are distributed to avoid a centralized queue [1, 14]. Our work is the first to examine the effect of immediate dispatching on throughput maximization.

We introduce a natural algorithm for the immediate dispatching model named FirstFit. In short, it fixes an ordering of the m machines $M_1, \ldots, M_m$, and assigns a newly-arrived job to the lowest-indexed machine that can feasibly accept it (the job is rejected if it is infeasible on all machines). We present the following two results regarding the analysis of FirstFit.

– For $m = 2$, we prove that FirstFit is $\frac{5}{3}$-competitive and that this is the best possible ratio for a deterministic algorithm with immediate dispatch. This places the model strictly between the immediate notification model (deterministic competitiveness $\frac{3}{2}$) and the immediate decision model (deterministic competitiveness $\frac{9}{5}$).
– For the case of unit-length jobs, we show that FirstFit has competitiveness $1/\left(1-\left(\frac{m-1}{m}\right)^m\right)$ for $m \geq 1$. Again, the model lies strictly between the others; an EDF strategy gives an optimal solution in the immediate notification model and our upper bound is less than a comparable lower bound with immediate decision for any m (both tend toward $\frac{e}{e-1} \approx 1.582$ as $m \to \infty$).

In addition, we present a variety of deterministic and randomized lower bounds for both the immediate dispatch and unrestricted models. Most notably, we strengthen the best-known lower bounds for the unrestricted and immediate notification models from $\frac{6}{5}$ to $\frac{5}{4}$ for the asymptotic case as $m \to \infty$. A summary of results for length $p \geq 3$ is given in Table 1, and for $p = 1$ in Table 2.

Previous Work. Baruah et al. consider an unrestricted model for scheduling jobs of varying length on a single machine to maximize the number of completed jobs, or the time spent on successful jobs [2]. Among their results, they prove that any reasonable nonpreemptive algorithm is 2-competitive with equal-length jobs, and that this is the best deterministic competitiveness. Specific 2-competitive algorithms are known for the unrestricted model [9], the immediate notification model [10], and the immediate decision model [6]. We note that for $m = 1$, the immediate notification and immediate dispatch models are the same, as any accepted job is trivially dispatched to the sole machine.

	Unrestricted/Immed. Notification				Immediate Dispatch				Immediate Decision			
	Randomized		Deterministic		Randomized		Deterministic		Randomized		Deterministic	
m	LB	UB	LB	UB	LB	UB	LB	UB	LB	UB	LB	UB
1	1.333 See [9]	1.667 See [5]	2 See [2, 9]	2 See [10]	1.333							2 See [6, 7]
2	**1.263** Thm. 6		1.5 See [7, 12]	1.5 See [7, 12]	**1.333** Thm. 3		**1.667** Thm. 5	**1.667** Thm. 1	1.333 See [6]		1.8 See [6]	1.8 See [6]
3	**1.256** Thm. 6		1.4 See [7]		**1.333** Thm. 3		**1.5** Thm. 4		1.333 See [6]		1.626 See [8]	1.730 See [6]
4	**1.255** Thm. 6		1.333 See [7]		**1.333** Thm. 3		**1.5** Thm. 5		1.333 See [6]		1.607 See [8]	1.694 See [6]
5	**1.25** Thm. 6		**1.333** Thm. 7		**1.333** Thm. 3		**1.429** Thm. 4		1.333 See [6]		1.599 See [8]	1.672 See [6]
6	**1.252** Thm. 6		1.3 See [7]		**1.333** Thm. 3		**1.444** Thm. 5		1.333 See [6]		1.594 See [8]	1.657 See [6]
7	**1.251** Thm. 6		**1.294** Thm. 7		**1.333** Thm. 3		**1.4** Thm. 4		1.333 See [6]		1.591 See [8]	1.647 See [6]
8	**1.251** Thm. 6		**1.308** Thm. 7		**1.333** Thm. 3		**1.417** Thm. 5		1.333 See [6]		1.589 See [8]	1.639 See [6]
∞	**1.25** Thm. 6		**1.25** Thm. 7		**1.333** Thm. 3		**1.333** Thms. 4–5		1.333 See [6]		1.582 See [8]	1.582 See [6]

Table 1. A summary of lower and upper bounds on the achievable competitiveness for this problem. Entries in bold are new results presented in this paper. All upper bounds apply for any p, while lower bounds generally require $p \geq 3$ (some apply for $p = 2$ or $p = 1$). Blank entries have bounds that are trivially extended from a neighboring model. For example, any upper bound with immediate decision applies to all models.

	m:	1	2	3	4	5	6	7	8	∞	citation
Immediate Notification	LB	1	1	1	1	1	1	1	1	1	
	UB	1	1	1	1	1	1	1	1	1	EDF
Immediate Dispatch	LB	1	**1.143**								Thm. 10
	UB	1	**1.333**	**1.421**	**1.463**	**1.487**	**1.504**	**1.515**	**1.523**	**1.582**	Thm. 2
Immediate Decision	LB	1	1.678	1.626	1.607	1.599	1.594	1.591	1.589	1.582	See [8]
	UB	1	1.8	1.73	1.694	1.672	1.657	1.647	1.639	1.582	See [6]

Table 2. A summary of deterministic lower and upper bounds for the special case of unit jobs (i.e., $p = 1$). Entries in bold are new results presented in this paper. We note that the upper bounds for the immediate decision model are the same as those given in Table 1 for general p.

With randomization, Goldman et al. show that no algorithm is better than $\frac{4}{3}$-competitive [9]. However, no algorithm with this ratio has (yet) been found. Chrobak et al. present a $\frac{5}{3}$-competitive randomized algorithm that is *barely random*, as it uses a single bit to choose between two deterministic strategies [5]. They also prove a lower bound of $\frac{3}{2}$ for such barely random algorithms.

For the two-machine version of the problem, Goldwasser and Pedigo [12], and independently Ding and Zhang [7], present a $\frac{3}{2}$-competitive deterministic algorithm in the immediate notification model, and a matching lower bound that applies even for the unrestricted model. Ding and Zhang also present a deterministic lower bound for $m \geq 3$ that approaches $\frac{6}{5}$ as $m \to \infty$.

The immediate decision model was first suggested by Ding and Zhang, and formally studied by Ding et al. [6]. They provide an algorithm named BestFit, defined briefly as follows. Jobs assigned to a given machine are committed to being executed in FIFO order. A newly-released job is placed on the most *heavily-loaded* machine that can feasibly complete it (or rejected, if none suffice). They prove that BestFit is $1/\left(1 - (\frac{m}{m+1})^m\right)$-competitive for any m. This expression equals 1.8 for $m = 2$ and approaches $\frac{e}{e-1} \approx 1.582$ as $m \to \infty$. They show that their analysis is tight for this algorithm, and they present a general lower bound for $m = 2$ and $p \geq 4$, showing that 1.8 is the best deterministic competitiveness for the immediate decision model. For $m \geq 3$, it is currently the best-known algorithm, even for the unrestricted model. Finally, they adapt the $\frac{4}{3}$ randomized lower bound for the unrestricted, single-processor case to the immediate decision model for $m \geq 1$. In subsequent work, Ebenlendr and Sgall prove that as $m \to \infty$, the 1.582 ratio of BestFit is the strongest possible for deterministic algorithms in the immediate decision model, even with unit-length jobs [8]. Specifically, they provide a lower bound of $\left(e^{\frac{m-1}{m}}\right) / \left(e^{\frac{m-1}{m}} - \frac{m}{m-1}\right)$.

Motivated by buffer management, Chin et al. consider scheduling *weighted* unit-length jobs to maximize the weighted throughput [4]. They give a randomized algorithm for a single processor that is 1.582-competitive. For multiprocessors, they give a $1/\left(1 - \left(\frac{m-1}{m}\right)^m\right)$-competitive deterministic algorithm for the unrestricted model. This is precisely our bound for FirstFit in the unweighted case with immediate dispatch, though the algorithms are not at all similar.

Although there is no previous work on maximizing throughput with immediate dispatch, Avrahami and Azar compare immediate dispatch to the unrestricted model for multiprocessor scheduling to minimize flow time or completion time [1]. For these metrics, once jobs are assigned to processors, each machine can schedule its jobs in FIFO order (and thus immediately assign time intervals).

Model and Notations. A scheduler manages $m \geq 1$ machines $M_1, \ldots M_m$. Job requests arrive, with job j specified by three nonnegative integer parameters: its release time r_j, its processing time

p_j, and its deadline d_j. We assume all processing times are equal, thus $p_j = p$ for a fixed constant p. To complete a job j, the scheduler must devote a machine to it for p consecutive time units during the interval $[r_j, d_j)$. We consider the nonpreemptive model so a running job cannot be interrupted.

The scheduler's goal is to maximize the number of jobs that are completed on time, and we use competitive analysis to measure the throughput of an online scheduling policy relative to the optimal schedule for a given instance [3, 13, 15]. In an online setting, we presume that a scheduler has no knowledge of job requests until the time at which the job is released. We do assume that the scheduler becomes aware of all of the job's parameters at that moment.

When several jobs have the same release time, there are two distinct models that can be studied. In the *online-list* model, such jobs are handled one at a time (in an order chosen by an adversary), with the scheduler making any required choices (e.g., immediate dispatch) for one job before learning of the next. However, the scheduler is assumed to have been presented with all releases at time t before having to decide what jobs to begin running at time t. In the *online-time* model, we assume that the scheduler is aware of the entire set of released jobs at a given time before making decisions about any of them. For most results in this paper, this distinction is irrelevant. Unless stated otherwise, we present algorithms and lower bound constructions that apply in either model.

Finally, we note the important distinction between having equal-length jobs and *unit-length* jobs (i.e., $p = 1$). With $p > 1$, the algorithm may start executing one job, only to learn of a new job that is released while the first is executing. In the unit-length model, this scenario is impossible.

2 The FirstFit Algorithm

We define an algorithm FirstFit as follows. Each machine maintains a queue of jobs that have been assigned to it but not yet completed. Let $Q_k(t)$ denote FirstFit's queue for M_k at the onset of time-step t (including any job that is currently executing). We define FirstFit so that it considers each arrival independently (the *online-list* model). To differentiate the changing state of the queues, we let $Q_k^j(t)$ denote the queue as it exists when job j with $r_j = t$ is considered. Note that $Q_k^j(t) \supseteq Q_k(t)$ may contain newly-accepted jobs that were considered prior to j. For a job j arriving at time t, we dispatch it to M_1 if $Q_1^j(t) \cup \{j\}$ remains feasible. Otherwise, we consider dispatching it to M_2, then M_3, and so on. If adding j is infeasible for each machine, it is rejected.

Unlike the BestFit algorithm for the immediate decision model [6], each machine can reorder its queue in FirstFit. In particular, when a machine becomes available, it begins the earliest-deadline job in its queue. Whether a job can be accepted onto a machine is also tested with the earliest-deadline first (EDF) schedule for the current queue plus the prospective job.

In the remainder of this section, we prove two key theorems about the performance of FirstFit. In Section 2.1, we show that FirstFit is $\frac{5}{3}$-competitive for equal-length jobs; this ratio is later shown to be optimal. In Section 2.2 we show that FirstFit is $1/\left(1 - \left(\frac{m-1}{m}\right)^m\right)$-competitive for the special case of unit-length jobs.

2.1 Optimal Competitiveness for Two Machines

We use an analysis style akin to that of [11, 12]. We fix a finite instance $\mathcal{I}$ and an optimal schedule Opt for that instance. Our analysis of the relative performance of FirstFit versus Opt is based upon two potential functions Φ^{FF} and Φ^{Opt} that measure the respective progress of the developing schedules over time. We analyze the instance by partitioning time into consecutive regions of the

form $[u, v)$ such that the increase in Φ^{FF} during a region is guaranteed to be at least that of Φ^{Opt}. Starting with $u = 0$, we end each region with the next time v at which the set $Q_1(v)$ can be feasibly scheduled on M_1 starting at time $v + p$ (as opposed to simply v). Such a time is well defined, as the queue eventually becomes empty and thus trivially feasible.

Before formalizing the potential functions, we must introduce the following notations. We let $S^{\text{FF}}(t)$ (resp. $S^{\text{Opt}}(t)$) denote the set of jobs started *strictly* before time t by FirstFit (resp. Opt). We define $D^{\text{FF}}(t) = S^{\text{Opt}}(t) \cap S^{\text{FF}}(\infty) \setminus S^{\text{FF}}(t)$ as the set of "delayed" jobs. These are started prior to time t by Opt, yet on or after time t by FirstFit. We define $D^{\text{Opt}}(t) = S^{\text{FF}}(t) \cap S^{\text{Opt}}(\infty) \setminus S^{\text{Opt}}(t)$ analogously. Lastly, we define a special set of "blocked" jobs for technical reasons that we will explain shortly. Formally, we let $B^{\text{Opt}}(t) \subseteq S^{\text{Opt}}(\infty) \setminus (S^{\text{Opt}}(t) \cup D^{\text{Opt}}(t))$, denote those jobs that were not started by either algorithm prior to t, but are started by Opt while FirstFit is still executing a job of $S^{\text{FF}}(t)$. Based on these sets, we define our potential functions as follows:

$$\begin{aligned} \Phi^{\text{FF}}(t) &= 5 \cdot |S^{\text{FF}}(t)| \quad + 2 \cdot |D^{\text{FF}}(t)| \\ \Phi^{\text{Opt}}(t) &= 3 \cdot |S^{\text{Opt}}(t)| + 3 \cdot |D^{\text{Opt}}(t)| + 2 \cdot |B^{\text{Opt}}(t)| \end{aligned}$$

Intuitively, these functions are payments to the respective schedules for work that is done. In the end, we will award 5 points to FirstFit for each job completed and 3 points to Opt, thus giving a $\frac{5}{3}$ competitive ratio. However, at intermediate times we award some advance payment for accepted jobs that are not yet started. For example, we award FirstFit with 2 points advanced credit if it has queued a job that Opt has already started. The algorithm gets the 3 additional points when it eventually starts that delayed job. In contrast, we immediately award Opt its full share of 3 credits for a delayed job. We will show that there are limited opportunities for Opt to carry a job from one region to the next as delayed, and we choose to pay for those discrepancies in advance.

The partial payment of 2 for jobs in $B^{\text{Opt}}(t)$ is a technical requirement related to our division of time into regions. The way we delimit regions guarantees that jobs FirstFit starts on M_1 complete by the end of a region. However, a job FirstFit starts on M_2 may execute past the region's end. Its could then hurt the algorithm's performance in the next region. We account for this problem by prepaying Opt during the analysis of the earlier region for progress made during the overhang.

Lemma 1. *If* FirstFit *rejects job j, it keeps all machines busy throughout the period $[r_j, d_j - p)$.*

Proof. If some M_k were idle at a time t, its queue is empty. Yet then it is feasible to add j to $Q_k^j(r_j)$, by scheduling that queue from $[r_j, t)$ as done by the algorithm, and then running j. □

Lemma 2. *Any job j started by* FirstFit *during a region $[u, v)$ has $d_j < v + p$, with the possible exception of the job started by M_1 at time u.*

Proof. For contradiction, assume there exists j with $d_j \geq v + p$. When j arrived at time r_j, the set $Q_1^j(r_j) \cup \{j\}$ was feasible since, given our definition of time v, we could use the algorithm's schedule until time v, followed by j during $[v, v + p)$ and $Q_1(v)$ starting at $v + p$. Therefore, such j must have been assigned to M_1. If j was started on M_1 at time $t > u$, $Q_1(t)$ could be feasibly scheduled starting at time $t + p$ by using the algorithm's schedule from $[t + p, v)$, running j from $[v, v + p)$, and the remaining $Q_1(v)$ starting at time $v + p$. However, the feasibility of $Q_1(t)$ starting at time $t + p$ contradicts our choice of v (rather than t) as the region's end. Therefore, no such job j exists. □

Lemma 3. *For a region $[u, v)$ in which M_1 idles at time u for* FirstFit, *$\Phi^{\text{FF}}(u) \geq \Phi^{\text{Opt}}(u)$ implies $\Phi^{\text{FF}}(v) \geq \Phi^{\text{Opt}}(v)$.*

Proof (sketch). In this case we define $v = u + 1$, noting that M_1 remains idle and so $Q_1(u+1) = \emptyset$ is trivially feasible. Any jobs started by Opt during the region must be ones that were previously started by FirstFit, and so we conclude that $\Phi^{\text{FF}}(v) = \Phi^{\text{FF}}(u)$ and $\Phi^{\text{Opt}}(v) = \Phi^{\text{Opt}}(u)$ □

Lemma 4. *For a region $[u, v)$ in which M_1 starts a job at time u for* FirstFit*, $\Phi^{\text{FF}}(u) \geq \Phi^{\text{Opt}}(u)$ implies $\Phi^{\text{FF}}(v) \geq \Phi^{\text{Opt}}(v)$.*

Proof (sketch). Let $n_1 \geq 1$ denote the number of jobs started by FirstFit on M_1 during the region, and $n_2 \geq 0$ denote the number of jobs started on M_2. Note that M_1 never idles during the region, for such a time would contradict our definition of v. Therefore, $v - u = p \cdot n_1$. We begin by considering possible contributions to $\Phi^{\text{Opt}}(v) - \Phi^{\text{Opt}}(u)$, partitioned as follows:

- $\mathbf{3 \cdot d}$ due to $d \geq 0$ jobs that are newly added to $D^{\text{Opt}}(v)$. Such delayed jobs must be started by FirstFit during the region yet held by Opt for a later region. By Lemma 2, there is at most one job started by FirstFit with expiration of v or later, thus $d \leq 1$.
- $\mathbf{3 \cdot a}$ due to $a \geq 0$ jobs that are newly added to $S^{\text{Opt}}(v)$, not previously credited as part of $D^{\text{Opt}}(u)$ or $B^{\text{Opt}}(u)$, and that were *accepted* by FirstFit upon their release. Given that these jobs were accepted by FirstFit and had not previously been started by Opt, they must either lie in $S^{\text{FF}}(v)$ or $D^{\text{FF}}(v)$.
- $\mathbf{3 \cdot r}$ due to $r \geq 0$ jobs that are newly added to $S^{\text{Opt}}(v)$, not previously credited as part of $B^{\text{Opt}}(u)$, and that were *rejected* by FirstFit upon their release.
- $\mathbf{1 \cdot b_{old}}$ due to $b_{\text{old}} \geq 0$ jobs that are newly added to $S^{\text{Opt}}(v)$ yet were previously credited as part of $B^{\text{Opt}}(u)$.
- $\mathbf{2 \cdot b_{new}}$ due to $b_{\text{new}} \geq 0$ jobs that newly qualify as blocked in $B^{\text{Opt}}(v)$. For such jobs to exist, there must be a newly-started job by FirstFit on M_2 whose execution extends beyond v. Since jobs have equal length, Opt can run at most one such blocked job per machine, thus $b_{\text{new}} \leq 2$.

Based on these notations, we have that $\Phi^{\text{Opt}}(v) - \Phi^{\text{Opt}}(u) = 3(d + a + r) + b_{\text{old}} + 2 \cdot b_{\text{new}}$. The remainder of our analysis depends upon the following two inequalities that relate Opt's progress to that of FirstFit.

- $\mathbf{2 \cdot n_1 \geq (a + r + b_{old})}$
  By definition, the jobs denoted by a, r, and b_{old} must be started by Opt strictly within the range $[u, v)$. There can be at most $2 \cdot n_1$ such jobs, given that the size of the region is known to be $v - u = p \cdot n_1$ and there are two machines.
- $\mathbf{2 \cdot n_2 \geq (r + b_{new})}$
  By definition, jobs denoted by b_{new} will be started by Opt at a time when FirstFit is still completing a job from the current region. That overhanging job of FirstFit must be on M_2. Furthermore, that job of FirstFit must be one denoted within n_2, for if it belongs to $S^{\text{FF}}(u)$, then Opt's job in question would belong to $B^{\text{Opt}}(u)$. Jobs denoted by r must be started at a time when M_2 is in use by FirstFit as per Lemma 1. Again, we rule out the possibility that M_2 for FirstFit was executing a job of $S^{\text{FF}}(u)$, as our definition of r excludes jobs that belong to $B^{\text{Opt}}(u)$. Given that all of the $r + b_{\text{new}}$ jobs of Opt start at times when FirstFit is running one of the n_2 newly-started jobs, and that at most one job of Opt can start per machine at a time when a particular job of FirstFit is running, we conclude that $(r + b_{\text{new}}) \leq 2 \cdot n_2$.

To complete the proof, we consider a case analysis depending on whether $n_1 - n_2 \geq d$. If so, we rely on additional $2(a + d)$ credits that can be claimed because jobs represented by a and d could

not have been partially credited within $D^{\text{FF}}(u)$. If $n_1 - n_2 < d$, it must be precisely that $n_1 = n_2$ and $d = 1$. Again, extra credits toward Φ^{FF} can be claimed by a further case analysis depending on whether $n_1 = 1$. A detailed argument is given in the Appendix. □

Theorem 1. *For $m = 2$, algorithm* FirstFit *is $\frac{5}{3}$-competitive.*

Proof. Initially, $\Phi^{\text{Opt}}(0) = \Phi^{\text{FF}}(0) = 0$. By repeatedly applying either Lemma 3 or 4 for regions $[u, v)$, we conclude that $\Phi^{\text{Opt}}(\infty) \leq \Phi^{\text{FF}}(\infty)$. Since there are no more delayed jobs at infinity, $D^{\text{FF}}(\infty) = D^{\text{Opt}}(\infty) = \emptyset$ and thus $3 \cdot |S^{\text{Opt}}(\infty)| \leq 5 \cdot |S^{\text{FF}}(\infty)|$, that is $\frac{\text{Opt}}{\text{FF}} \leq \frac{5}{3}$. □

2.2 Unit-length Jobs

We consider a job j to be *regular* with respect to FirstFit if the machine to which it is dispatched (if any) never idles during the interval $[r_j, d_j)$. We consider an instance $\mathcal{I}$ to be *regular* with respect to FirstFit if all jobs are regular.

Lemma 5. *For $p = 1$, the worst case competitive ratio for* FirstFit *occurs on a regular instance.*

Proof. Consider an irregular instance $\mathcal{I}$, and let j on M_k be the last irregular job started by FirstFit. Let s_j denote the time at which j starts executing. The idleness of M_k leading to j's irregularity cannot occur while j is in the queue, so it must occur within the interval $[s_j + 1, d_j)$. We claim that for $r_j \leq t \leq s_j$, $j \in Q_k(t)$ has the largest deadline of jobs in the queue. For the sake of contradiction, assume jobs j and j' are in the queue at some point, for a j' coming after j in EDF ordering. Job j' must also be irregular, since we know there is idleness within interval $[s_j + 1, d_j) \subseteq [r_{j'}, d_{j'})$. Since j' starts after j by EDF, this contradicts our choice of j as the last irregular job to be started.

We claim that FirstFit produces the exact schedule for $\mathcal{I}' = \mathcal{I} - \{j\}$ as it does for $\mathcal{I}$, except replacing j by an idle slot. In essence, we argue that j's existence never affects the treatment of other jobs. Since j always has a deadline that is at least one greater than the cardinality of Q_k while in the queue, it cannot adversely affect a feasibility test when considering the dispatch of another job to M_k. Also, since j has the largest deadline while in Q_k, its omission does not affect the choice of jobs that are started, other than by the time s_j when it is the EDF job, and therefore $Q_k(s_j) = \{j\}$. There are no other jobs to place in the time slot previously used for j.

To conclude, since FirstFit completes one less job on $\mathcal{I}'$ than $\mathcal{I}$, and Opt loses at most one job, the competitive ratio on $\mathcal{I}'$ is at least as great as on $\mathcal{I}$. □

Theorem 2. *For $p = 1$, algorithm* FirstFit *is $\dfrac{1}{1 - \left(\frac{m-1}{m}\right)^m}$-competitive.*

Proof. By Lemma 5, we can prove the competitiveness of FirstFit by analyzing an arbitrary *regular* instance. We rely on a charging scheme inspired by the analysis of BestFit in the immediate decision model [6], but with a different sequence of charges. We define $Y_k = (m-1)^{m-k} \cdot m^{k-1}$ for $1 \leq k \leq m$. We note that $\sum_{k=1}^{m} Y_k = m^m - (m-1)^m$ is a geometric sum with the common ratio $\frac{m}{m-1}$. A job i started at time t by Opt will distribute $m^m - (m-1)^m$ units of charge by assigning $Y_1, Y_2, \ldots Y_k$ respectively to the jobs $j_1, j_2, \ldots, j_k$ run by FirstFit at time t on machines $M_1, M_2, \ldots, M_k$ for some k. When $k < m$, the remaining charge of $\sum_{z=k+1}^{m} Y_z$ is assigned to i itself; this is well-defined, as i must have been accepted by FirstFit given that there is an idle machine at time t when i is feasible.

We complete our proof by showing that each job j run by FirstFit collects at most m^m units of charge, thereby proving the competitiveness of $\frac{m^m}{m^m-(m-1)^m} = \frac{1}{1-\left(\frac{m-1}{m}\right)^m}$. Consider a job j that is run by FirstFit on M_k. By our definition of regularity, machine M_k (and hence machines M_1 through M_{k-1} by definition of FirstFit) must be busy at a time when Opt starts j. Therefore, j receives at most $\sum_{z=k+1}^{m} Y_z$ units of supplemental charge from itself. In addition, j may collect up to $m \cdot Y_k$ from the jobs that Opt runs at the time FirstFit runs j. So j collects at most $m \cdot Y_k + \sum_{z=k+1}^{m} Y_z = (m-1) \cdot Y_k + \sum_{z=k}^{m} Y_z$. We prove by induction on k that $(m-1) \cdot Y_k + \sum_{z=k}^{m} Y_z = (m-1) \cdot Y_1 + \sum_{z=1}^{m} Y_z$. This is trivially so for $k = 1$. For $k > 1$, $(m-1) \cdot Y_k = (m-1)^{m-(k-1)} \cdot m^{k-1} = m \cdot Y_{k-1}$. Therefore $(m-1) \cdot Y_k + \sum_{z=k}^{m} Y_z = m \cdot Y_{k-1} + \sum_{z=k}^{m} Y_z = (m-1) \cdot Y_{k-1} + \sum_{z=k-1}^{m} Y_k$, which by induction equals $(m-1) \cdot Y_1 + \sum_{z=1}^{m} Y_z$. Finally, we note that $(m-1) \cdot Y_1 = (m-1)^m$ and $\sum_{z=1}^{m} Y_z = m^m - (m-1)^m$ so therefore each job j run by FirstFit collects at most m^m units of charge. □

We note that our analysis of FirstFit is tight, as demonstrated by the following instance. Consider $m+1$ "waves" of jobs. For $1 \leq w \leq m$, wave w is $m \cdot Y_{m+1-w}$ jobs with release time $\sum_{z=1}^{w-1} Y_{m+1-z}$ and deadline m^m. The final wave is $m \cdot (m-1)^m$ jobs with release time $m^m - (m-1)^m$ and deadline m^m. FirstFit assigns the first wave to M_1, the second to M_2, and so on. Each wave uses its machine until time m^m. Thus FirstFit must reject the final $m(m-1)^m$ jobs and runs a total of $m \cdot \sum_{k=1}^{m} Y_k = m(m^m - (m-1)^m)$ jobs. In contrast, Opt can run all $m \cdot m^m$ jobs by distributing each wave across all m machines. This leads to a competitive ratio of $\frac{m^m}{m^m-(m-1)^m}$.

3 Lower Bounds

In this section, we provide lower bounds on the competitiveness of randomized and deterministic algorithms for the immediate dispatch model, the unrestricted model, and the special case of $m = 2$ and $p = 1$. In our constructions, we use the notation $\langle r_j, d_j \rangle$ to denote a job with release time r_j and deadline d_j. Goldman et al. provide a prototypical $\frac{4}{3}$-competitive lower bound for randomized algorithms on a single machine in the unrestricted model [9]. While that construction cannot be directly applied to the multiple machine case in the unrestricted model, Ding et al. use such a construction in the *immediate decision* model to provide a randomized lower bound of $\frac{4}{3}$ for any number of machines [6]. We show this lower bound applies to the *immediate dispatch* model as well.

Theorem 3. *For the* immediate dispatch *model with $p \geq 2$, a randomized algorithm cannot have competitiveness strictly better than $\frac{4}{3}$ against an oblivious adversary.*

Proof. Formally, we prove a lower bound on the randomized competitiveness by applying Yao's principle [3, 16] and bounding the expected value of a deterministic algorithm against the following random distribution. We consider two instances. Both begin with m jobs having parameters $\langle 0, 2p+1 \rangle$. For a fixed deterministic algorithm, we let α denote the number of machines that have been assigned two jobs from the first group or one job that it chose to start at time 0.

Our first instance continues with m jobs having parameters $\langle p, 2p \rangle$. The $m - \alpha$ machines that did not have two initial jobs assigned nor start an initial job at time 0 are able to run at most one job each. Combining that with at most 2 jobs for each of the other α machines, the online algorithm runs at most $2 \cdot \alpha + (m - \alpha) = m + \alpha$ jobs, for a competitive ratio of at least $\frac{2m}{m+\alpha}$ on this instance. Our second instance continues with m jobs having parameters $\langle 1, p+1 \rangle$. At least α of the second wave of jobs must be rejected, since none can be scheduled on the α machines that are otherwise committed. This leads to a competitive ratio of at least $\frac{2m}{2m-\alpha}$ against the second instance.

For a uniform random distribution over these two instances, a deterministic algorithm has expected competitive ratio at least $\frac{1}{2}\left(\frac{2m}{m+\alpha}+\frac{2m}{2m-\alpha}\right)$. This is minimized at $\frac{4}{3}$ when $\alpha=\frac{m}{2}$. Therefore, any randomized algorithm is at best $\frac{4}{3}$-competitive against an oblivious adversary. □

We now prove slightly stronger bounds for *deterministic* algorithms. In Theorem 4, we exploit that an algorithm cannot precisely choose $\alpha=\frac{m}{2}$ when m is odd. In Theorem 5, we give a job with large deadline and, once the algorithm starts it, overlay a construction similar to above.

Theorem 4. *For the* immediate dispatch *model with $p\geq 2$ and m odd, a deterministic algorithm cannot have competitiveness strictly better than $\frac{4m}{3m-1}$.*

Proof (sketch). We consider the same two instances as in the proof of Theorem 3, for which a deterministic algorithm has a competitive ratio of at least $\max(\frac{2m}{m+\alpha},\frac{2m}{m-\alpha})$. For odd m, this is minimized with $\alpha=\lfloor\frac{m}{2}\rfloor=\frac{m-1}{2}$ or $\lceil\frac{m}{2}\rceil=\frac{m+1}{2}$. In either case, the competitive ratio is lower bounded by $\max(\frac{4m}{3m-1},\frac{4m}{3m+1})=\frac{4m}{3m-1}$. □

Theorem 5. *For the* immediate dispatch *model with $p\geq 3$ and m even, a deterministic algorithm cannot have competitiveness strictly better than $\frac{4m+2}{3m}$.*

Proof. Our adversary begins by presenting a single job with parameter $\langle 0,4p+1\rangle$. For an arbitrary deterministic algorithm, let t be the time at which this job is started if no other jobs were to arrive. We next introduce a set of m identical jobs with parameters $\langle t+1,t+2p+2\rangle$. Let α denote the number of machines at time $t+1$ that are either running a job or have two jobs already assigned. Our adversary will present one of two possible continuations.

In the first, a set of m additional jobs are released with parameters $\langle t+p+1,t+2p+1\rangle$. Opt runs all $1+2m$ jobs, while the algorithm gets at most $m+\alpha$, running at most 2 jobs on the α machines and at most 1 on the others. Our second instance begins with the same $1+m$ jobs as the first. It continues with m jobs with parameters $\langle t+2,t+p+2\rangle$. Again, Opt runs $1+2m$ jobs, but now the algorithm gets at most $1+2m-\alpha$ since it has to reject at least α of the final batch, given the α machines with conflicting commitments. For fixed α, an adversary chooses the worst of these for a lower bound of $\max(\frac{1+2m}{m+\alpha},\frac{1+2m}{1+2m-\alpha})$. This is minimized at $\frac{4m+2}{3m}$ when $\alpha=\lfloor\frac{1+m}{2}\rfloor=\frac{m}{2}$. □

Although the $\frac{4}{3}$-competitive lower bound construction for the single-machine case has been adapted to the multiple machine case in the immediate decision and immediate dispatch models, it does not directly apply to the less restrictive model of immediate notification or the original unrestricted model. If facing the construction used in Theorem 3, an optimal deterministic algorithm could accept the initial m jobs with parameters $\langle 0,2p+1\rangle$, starting $\frac{m}{3}$ of them at time 0 and centrally queuing the other $\frac{2m}{3}$. If at time 1 it faces the arrival of m additional jobs with parameters $\langle 1,p+1\rangle$, it can accept $\frac{2m}{3}$ of them on idle machines, while still completing the remaining initial jobs at time $p+1$ on those machines. The competitive ratio in this setting is $2m/(m+\frac{2m}{3})=\frac{6}{5}$. If no jobs arrive by time 1 for the given adversarial construction, it can commit another $\frac{m}{3}$ of the machines to run initial jobs from $[1,p+1)$, with the final third of the initial jobs slated on those same machines from $[p+1,2p+1)$. In that way, it retains room for $\frac{2m}{3}$ jobs in a second wave during the interval $[p,2p]$, by using the idle machines and the first third of the machines that will have completed their initial job, again leading to a competitive ratio of $\frac{6}{5}$. Ding and Zhang [7] provide a slightly stronger deterministic bound for fixed values of m, by releasing a single initial job with larger deadline, followed by the classic construction (akin to our construction from Theorem 5).

In our next series of results, we give a new construction that strengthens the randomized and deterministic lower bounds for these models, showing that competitiveness better than $\frac{5}{4}$ is impossible in general. By increasing the number of jobs in the second wave of one of the instances to $2m$ jobs from m, we reduce the "flexibility benefit" from leaving machines idle at time 0. This changes the balancing point between the instances and increases the competitive ratio.

Theorem 6. *For the* unrestricted *model with $p \geq 2$, a randomized algorithm cannot have competitiveness strictly better than the following:*

$m \bmod 5$	0	1	2	3	4
competitive ratio	$\frac{5}{4}$	$\frac{20m^2}{16m^2-1}$	$\frac{30m^2}{24m^2-1}$	$\frac{30m^2}{24m^2-1}$	$\frac{20m^2}{16m^2-1}$

Proof (sketch). We apply Yao's principle with a distribution of two possible instances. Both instances begin with m jobs with parameters $\langle 0, 2p+1\rangle$. In analyzing a particular deterministic algorithm, we let α denote the number of machines that begin running a job at time 0.

Our first instance continues with $2m$ jobs having parameters $\langle p, 3p\rangle$. Opt runs all $3m$ jobs by scheduling the initial batch of jobs from $[0, p)$ and the final batch from $[p, 3p)$. For the online algorithm, a machine that does not starting a job at time 0 can run at most 2 jobs. Therefore, it runs at most $3\alpha + 2 \cdot (m - \alpha) = 2m + \alpha$ jobs, for a competitive ratio of at most $\frac{3m}{2m+\alpha}$ on this instance. Our second instance continues with m jobs with parameters $\langle 1, p+1\rangle$. Opt runs all $2m$ jobs by scheduling the second batch of jobs from $[1, p+1)$ and the initial batch from $[p+1, 2p+1)$. In contrast, the online algorithm runs at most $2m - \alpha$ jobs, as it must reject α of the jobs arriving at time 1. Thus, its competitive ratio is at most $\frac{2m}{2m-\alpha}$ on this instance.

For $m \equiv 0 \pmod 5$, we select the first instance with probability $\frac{1}{2}$. The expected competitive ratio of a deterministic algorithm for this distribution is at most $\frac{1}{2}\left(\frac{3m}{2m+\alpha} + \frac{2m}{2m-\alpha}\right)$. This expression is minimized at $\frac{5}{4}$ by an algorithm choosing $\alpha = \frac{2m}{5}$. Since all deterministic algorithms have expected competitiveness at least $\frac{5}{4}$ against this distribution, any randomized algorithm is at best $\frac{5}{4}$-competitive against an oblivious adversary. For other modularities of m, an even stronger bound holds because the algorithm cannot choose $\alpha = \frac{2m}{5}$. The details are deferred to the Appendix. □

Our next theorem strengthens the bound for deterministic algorithms by first releasing a single job with large deadline (as we did with Theorem 5 for the immediate assignment model).

Theorem 7. *For the* unrestricted *model with $p \geq 3$, a deterministic algorithm cannot have competitiveness strictly better than the following:*

$m \bmod 5$	0	1	2	3	4
competitive ratio	$\frac{5}{4}\left(1 + \frac{1}{3m}\right)$	$\frac{5}{4}\left(1 + \frac{1}{(4m+1)}\right)$	$\frac{5}{4}\left(1 + \frac{3}{(12m+1)}\right)$	$\frac{5}{4}\left(1 + \frac{3}{(8m+1)}\right)$	$\frac{5}{4}\left(1 + \frac{1}{(4m-1)}\right)$

Proof (sketch). Our adversary begins by presenting a single job with parameter $\langle 0, 5p-1\rangle$. For an arbitrary deterministic algorithm, let t be when the job is started if no others arrive. Next, a set of m' identical jobs with parameters $\langle t+1, t+2p+2\rangle$ arrive, where $m' = m-1$ if $m = 4 \pmod 5$ and $m' = m$ otherwise. Let α denote the number of jobs (including the original job) started by the online algorithm on or before time $t+1$. Our adversary presents one of two possible continuations. In the first, a set of $2m$ new jobs with parameters $\langle t+p+1, t+3p+1\rangle$ are released, giving a competitive

ratio of at least $\frac{1+m'+2m}{2m+\alpha}$. In the second, m new jobs with parameters $\langle t+2, t+p+2\rangle$ arrive, giving a competitive ratio of at least $\frac{1+m'+m}{1+m'+m-\alpha}$. The precise lower bounds come from analyzing the optimal choice of α for varying values of m; we defer details to the Appendix. □

The construction of Theorem 7 requires $p \geq 3$, in order to leverage the introduction of the initial job $\langle 0, 5p-1\rangle$. For $p = 2$, we provide slightly weaker bounds as follows.

Theorem 8. *For the* unrestricted *model with $p = 2$, a deterministic algorithm cannot have competitiveness strictly better than the following:*

m mod 5	1	2	3	4
competitive ratio	$\frac{15m}{12m-2}$	$\frac{10m}{8m-1}$	$\frac{15m}{12m-1}$	$\frac{5m}{4m-1}$

Proof. Construction from Theorem 6, with deterministic algorithm choosing α as $\lfloor \frac{2m}{5} \rfloor$ or $\lceil \frac{2m}{5} \rceil$. □

Finally, we focus on the special case of $p = 1$ and $m = 2$. Our analysis in Section 2.2 shows that FirstFit is precisely $\frac{4}{3}$-competitive in this setting. However, the $\frac{4}{3}$ lower bounds from the previous theorems do not apply with $p = 1$; an adversary cannot force the rejection of new jobs due to machines that are committed to other tasks. With the following theorems, we provide alternative lower bounds (albeit, weaker) for the unit-length job, drawing a distinction between the *online-time* and *online-list* models, as defined in the introduction.

Theorem 9. *For the immediate dispatch model with $p = 1$ and $m = 2$, a deterministic online-time algorithm cannot have a constant competitiveness ratio strictly better than* 9/8.

Proof. We first prove that the competitive ratio is at least 10/9 and then show how to repeat part of the instance to give the claimed bound. For the 10/9 bound, the instance begins with five jobs: two jobs with parameters $\langle 0, 1\rangle$, $A = \langle 0, 2\rangle$, $B = \langle 0, 3\rangle$, and $C = \langle 0, 5\rangle$. To be competitive, the algorithm must accept all five jobs, dispatching a $\langle 0, 1\rangle$ job to each machine and, without loss of generality, job A to M_1. We have the following cases:

Case 1: B assigned to M_1: Two jobs $\langle 2, 3\rangle$ arrive. The algorithm cannot accept them both since M_1 is busy with A and B; Opt runs A and B at time 1, the $\langle 2, 3\rangle$ jobs at time 2, and C at time 3. Competitive ratio is at best 7/6.

Case 2: B and C assigned to M_2: A job $\langle 1, 2\rangle$ and four jobs $\langle 3, 5\rangle$ arrive. The algorithm can accept at most four of those five, achieving at most 9 of 10. Opt runs all, with A and the $\langle 1, 2\rangle$ job at time 1, B and C at time 2, and the four $\langle 3, 5\rangle$ jobs starting at time 3.

Case 3: C assigned to M_1, B assigned to M_2: A job $\langle 1, 2\rangle$ and job $D = \langle 1, 4\rangle$ arrive. To be competitive, the algorithm must accept both, starting A and the $\langle 1, 2\rangle$ job at time 1. At time 2, jobs B, C, and D remain. Job B must start immediately. We have one of the following:

- **Case 3a: D assigned to M_1:** A job $\langle 2, 3\rangle$ and two jobs $\langle 4, 5\rangle$ arrive. The algorithm is unable to accept all ten jobs. With B on M_2, the new $\langle 2, 3\rangle$ job must run on M_1. Yet then C and D occupy M_1 until time 5, and one of the $\langle 4, 5\rangle$ jobs is rejected. Opt can run A and $\langle 1, 2\rangle$ at time 1, B and $\langle 2, 3\rangle$ at time 2, C and D at time 3, and the two $\langle 4, 5\rangle$ jobs at time 4.
- **Case 3b: D assigned to M_2:** Two jobs $\langle 3, 4\rangle$ arrive. The algorithm cannot accept both since M_2 must run B and D by time 4. Opt can run A and $\langle 1, 2\rangle$ at time 1, B and D at time 2, the two $\langle 3, 4\rangle$ jobs at time 3, and C at time 4.

The lowest ratios come from cases 2 and 3a, with 9 of 10 jobs accepted. At time 4 in those cases, both machines are running jobs with deadline 5. Then we repeat the construction using those jobs in place of the pair $\langle 0, 1\rangle$. That is, we release $A_2 = \langle 4, 6\rangle$, $B_2 = \langle 4, 7\rangle$, $C_2 = \langle 4, 9\rangle$, and $D_2 = \langle 5, 8\rangle$ (if needed) to force at least 2 rejections of out of at most 18 jobs, for a bound of $18/16 = 9/8$. □

Theorem 10. *For the immediate dispatch model with $p = 1$ and $m = 2$, a deterministic online-list algorithm cannot have competitiveness strictly better than $8/7$.*

4 Conclusions

The main contribution of this paper has been the introduction of the *immediate dispatch* model for the problem of maximizing throughput with equal-length jobs. We demonstrate that this model is strictly more difficult than the *immediate notification* model, and strictly easier than the *immediate decision* model. We hope that an understanding of these models may help in settling the primary open problem in this area, namely to develop stronger algorithms for $m \geq 3$ in any of these models.

References

1. N. Avrahami and Y. Azar. Minimizing total flow time and total completion time with immediate dispatching. *Algorithmica*, 47(3):253–268, 2007.
2. S. K. Baruah, J. R. Haritsa, and N. Sharma. On-line scheduling to maximize task completions. *J. Combin. Math. and Combin. Computing*, 39:65–78, 2001.
3. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, New York, 1998.
4. F. Y. L. Chin, M. Chrobak, S. P. Y. Fung, W. Jawor, J. Sgall, and T. Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. *J. Discrete Algorithms*, 4(2):255–276, 2006.
5. M. Chrobak, W. Jawor, J. Sgall, and T. Tichý. Online scheduling of equal-length jobs: Randomization and restarts help. *SIAM J. Comput.*, 36(6):1709–1728, 2007.
6. J. Ding, T. Ebenlendr, J. Sgall, and G. Zhang. Online scheduling of equal-length jobs on parallel machines. In L. Arge and M. Hoffmann, editors, *Proc. 15th European Symp. on Algorithms (ESA)*, volume 4698 of *Lecture Notes in Computer Science*, pages 427–438, Eilat, Israel, Oct. 2007. Springer-Verlag.
7. J. Ding and G. Zhang. Online scheduling with hard deadlines on parallel machines. In *Proc. Second Int. Conference on Algorithmic Aspects in Information and Management*, volume 4041 of *Lecture Notes in Computer Science*, pages 32–42, Hong Kong, China, June 2006. Springer-Verlag.
8. T. Ebenlendr and J. Sgall. A lower bound for scheduling of unit jobs with immediate decision on parallel machines. In E. Bampis and M. Skutella, editors, *Proc. Sixth Workshop on Approximation and Online Algorithms (WAOA)*, Lecture Notes in Computer Science, pages 43–52, Germany, Sept. 2008. Springer-Verlag.
9. S. Goldman, J. Parwatikar, and S. Suri. On-line scheduling with hard deadlines. *J. Algorithms*, 34(2):370–389, Feb. 2000.
10. M. H. Goldwasser and B. Kerbikov. Admission control with immediate notification. *J. Scheduling*, 6(3):269–285, May/June 2003.
11. M. H. Goldwasser and A. B. Misra. A simpler competitive analysis for scheduling equal-length jobs on one machine with restarts. *Information Processing Letters*, 107(6):240–245, Aug. 2008.
12. M. H. Goldwasser and M. Pedigo. Online nonpreemptive scheduling of equal-length jobs on two identical machines. *ACM Trans. on Algorithms*, 5(1), Nov. 2008.
13. A. Karlin, M. Manasse, L. Rudolph, and D. Sleator. Competitive snoopy paging. *Algorithmica*, 3(1):70–119, 1988.
14. K. Pruhs. Competitive online scheduling for server systems. *SIGMETRICS Perform. Eval. Rev.*, 34(4):52–58, 2007.
15. D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28:202–208, 1985.
16. A. C.-C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. 18th Symp. on Foundations of Computer Science (FOCS)*, pages 222–227, Providence, Rhode Island, Oct. 31–Nov. 2, 1977.

A Omitted proofs

Proof (of Lemma 3). For M_1 to be idle at time u, its queue must be empty. Furthermore, there must not be any new arrivals at time u. In this case, $v = u + 1$, as $Q_1(u+1)$ remains empty (as it does not include jobs released at $u + 1$) and thus trivially feasible. The second machine may still be processing a job that started at an earlier time, but other than that job, its queue must be empty as well. This is because any job that could feasibly be started after time t would have been assigned to M_1, given its idleness at time u. As no jobs are started by FirstFit during the region, $S^{\mathrm{FF}}(v) = S^{\mathrm{FF}}(u)$, and since the algorithm's queues do not contain any jobs with expiration on or after u, $D^{\mathrm{FF}}(v) = D^{\mathrm{FF}}(u) = \emptyset$. We conclude that $\Phi^{\mathrm{FF}}(v) = \Phi^{\mathrm{FF}}(u)$.

Because FirstFit does not start any new jobs during the region, it is impossible for Opt to get credit for any newly-delayed or newly-blocked jobs. The only other potential contribution toward Φ^{Opt} during this region would be from jobs that Opt starts. However, any job that Opt starts during this region must already lie in $D^{\mathrm{Opt}}(u)$, and thus be credited in $\Phi^{\mathrm{Opt}}(u)$. This is so because a job j started by Opt satisfies $r_j \le u \le d_j - p$. By Lemma 1, it must have been accepted by FirstFit. Since there are no jobs with such an expiration in the algorithm's queues, it must have been started strictly before u and thus in $D^{\mathrm{Opt}}(u)$. We conclude that $\Phi^{\mathrm{Opt}}(v) = \Phi^{\mathrm{Opt}}(u)$. □

Proof (continuation of Lemma 4). We show that $\Phi^{\mathrm{FF}}(v) - \Phi^{\mathrm{FF}}(u) \ge \Phi^{\mathrm{Opt}}(v) - \Phi^{\mathrm{Opt}}(u)$, and thus that $\Phi^{\mathrm{FF}}(u) \ge \Phi^{\mathrm{Opt}}(u)$ implies $\Phi^{\mathrm{FF}}(v) \ge \Phi^{\mathrm{Opt}}(v)$. We consider two possible cases, the first of which is when $n_1 - n_2 - d \ge 0$. We begin by noting that every job started by FirstFit accounts for at least 3 units of credit toward Φ^{FF}, as it is newly added to $S^{\mathrm{FF}}(v)$, yet possibly credited within $D^{\mathrm{FF}}(u)$. We claim an additional $2(a + d)$ units of credit toward Φ^{FF}. Notice that each of the jobs denoted by a and d have been accepted by FirstFit yet not started prior to u. These jobs could not have been partially credited within $D^{\mathrm{FF}}(u)$, as that would imply containment in $S^{\mathrm{Opt}}(u)$, contradicting the definition of a and d. Therefore each of these jobs either becomes newly delayed by FirstFit, thus contributing 2 as a member of $D^{\mathrm{FF}}(v)$, or is in $S^{\mathrm{FF}}(v) \setminus D^{\mathrm{FF}}(u)$ and thus contributes a full 5 units rather than our previously assumed net gain of 3. Therefore, we deduce the following.

$$
\begin{aligned}
\Phi^{\mathrm{FF}}(v) - \Phi^{\mathrm{FF}}(u) &\ge 3(n_1 + n_2) + 2(a + d) \\
&= 3(n_1 + n_2) + 2(a + d + r) - 2(r + b_{\mathrm{new}}) + 2 \cdot b_{\mathrm{new}} \\
&\ge 3(n_1 + n_2) + 2(a + d + r) - 4 \cdot n_2 + 2 \cdot b_{\mathrm{new}} && \text{as } 2 \cdot n_2 \ge (r + b_{\mathrm{new}}) \\
&= 2 \cdot n_1 + (n_1 - n_2) + 2(a + d + r) + 2 \cdot b_{\mathrm{new}} \\
&\ge (a + r + b_{\mathrm{old}}) + (n_1 - n_2) + 2(a + d + r) + 2 \cdot b_{\mathrm{new}} && \text{as } 2 \cdot n_1 \ge (a + r + b_{\mathrm{old}}) \\
&= 3(a + d + r) - d + b_{\mathrm{old}} + (n_1 - n_2) + 2 \cdot b_{\mathrm{new}} \\
&\ge 3(a + d + r) + b_{\mathrm{old}} + 2 \cdot b_{\mathrm{new}} && \text{assuming } n_1 - n_2 - d \ge 0 \\
&= \Phi^{\mathrm{Opt}}(v) - \Phi^{\mathrm{Opt}}(u)
\end{aligned}
$$

It remains to discuss the case when $n_1 - n_2 - d < 0$. Given that $n_1 \ge n_2$ and $d \le 1$, this inequality implies that $n_1 = n_2$ and $d = 1$. We begin by considering the first job executed by FirstFit on M_1 (that contributing to $d = 1$ for Opt). Since this job is delayed by Opt it cannot belong to $D^{\mathrm{FF}}(u)$ and therefore contributes a new credit of 5 toward Φ^{FF} rather than 3. Furthermore, Opt's delay of the job denoted by d ensures that its deadline is at least $v + p$. By Lemma 2, all other jobs run by FirstFit in the region have deadline strictly before $v + p$. Had any of those jobs been released or or before u, they would have been feasible to dispatched to M_1 and, given the EDF scheduling

policy, they would have been started rather than the job denoted by d. Therefore, all jobs started by FirstFit during this region, were released at time u or later and cannot belong to $D^{\text{FF}}(u)$. As a result, each contributes 5 credits and $\Phi^{\text{FF}}(v) - \Phi^{\text{FF}}(u) \geq 5(n_1 + n_2) = 10 \cdot n_1$, as $n_1 = n_2$.

We conclude by considering two further subcases. First, we consider when $n_1 \geq 2$. In this case, we trivially have that $4 \cdot n_1 > 3 \cdot d + 2 \cdot b_{\text{new}}$ as $d = 1$ and $b_{\text{new}} \leq 2$. With that inequality, we see

$$\begin{aligned}
\Phi^{\text{FF}}(v) - \Phi^{\text{FF}}(u) &\geq 10 \cdot n_1 \\
&\geq 3(a + r + b_{\text{old}}) + 4 \cdot n_1 && \text{as } 6 \cdot n_1 \geq 3 \cdot (a + r + b_{\text{old}}) \\
&= 3(a + r + d) + 3 \cdot b_{\text{old}} - 3 \cdot d + 4 \cdot n_1 \\
&\geq 3(a + r + d) + b_{\text{old}} - 3 \cdot d + 4 \cdot n_1 \\
&> 3(a + r + d) + b_{\text{old}} + 2 \cdot b_{\text{new}} && \text{as } 4 \cdot n_1 - 3 \cdot d > 2 \cdot b_{\text{new}} \\
&= \Phi^{\text{Opt}}(v) - \Phi^{\text{Opt}}(u)
\end{aligned}$$

The final scenario is when $n_1 = n_2 = d = 1$. In this case, the algorithm starts one job, denoted as j, on M_1 at time u and another job, denoted as k, on M_2 at some time $u \leq t < v$. By our earlier argument, $\Phi^{\text{FF}}(v) - \Phi^{\text{FF}}(u) \geq 5(n_1 + n_2) = 10$. In contrast, we consider contributions toward Φ^{Opt}. The delayed job j results in 3 units of credit for Opt. This leaves us with a net surplus of 7 units in favor of FirstFit thus far in our analysis. All remaining credits towards Φ^{Opt} must be due to jobs started by Opt either during this region or blocked by k's overhang into $[v, t + p)$. If Opt started only one job per machine in this analysis, it would receive at most 3 additional credits for each, and therefore $\Phi^{\text{Opt}}(v) - \Phi^{\text{Opt}}(u) \leq 9 < 10 = \Phi^{\text{FF}}(v) - \Phi^{\text{FF}}(u)$. For Opt to get credit for two different jobs on the same machine, it must start one job during $[u, t)$ and the other during $[v, t + p)$. The latter of those two will only be credited 2 towards Φ^{Opt} as a new member of $B^{\text{Opt}}(v)$ (note that k cannot itself be held as delayed by Opt given that it expires strictly before v). If a job started by Opt during $[u, t)$ were a member of $B^{\text{Opt}}(u)$, it would only produce a net gain of 1 which, combined with the latter job's 2, nets Opt with only 3 due to work on that machine.

To surpass a gain of 3 per machine, Opt must receive credit for starting a non-blocked job during the range $[u, t)$. That would produce a gain on that machine of 5 (with 3 for the first job and 2 for the second). A *non-blocked* job started by Opt during $[u, t)$ must start after FirstFit completes any jobs hanging from $S^{\text{FF}}(u)$ on M_2, yet before FirstFit starts k. M_2 is idle during such an interim, so Lemma 1 implies that the job started by Opt is accepted by FirstFit. Since Opt receives new credit for the job, it was not previously scheduled by FirstFit. Furthermore, that job cannot be k itself, because if k had already been released and assigned to M_2 by FirstFit, that machine would not be idle. Therefore, the job in question that produces additional gain for Opt must be newly added to $D^{\text{FF}}(v)$. This results in an additional gain of 2 toward Φ^{FF}, beyond the 10 we have previously described. This counteracts the fact that Opt received a gain of 5 rather than 3, and so we have that $\Phi^{\text{Opt}}(v) - \Phi^{\text{Opt}}(u) = 11$ and $\Phi^{\text{FF}}(v) - \Phi^{\text{FF}}(u) = 12$. If Opt were to do the same on both machines, we have that $\Phi^{\text{Opt}}(v) - \Phi^{\text{Opt}}(u) = 13$ yet $\Phi^{\text{FF}}(v) - \Phi^{\text{FF}}(u) = 14$. In all cases, we conclude that $\Phi^{\text{Opt}}(v) - \Phi^{\text{Opt}}(u) \leq \Phi^{\text{FF}}(v) - \Phi^{\text{FF}}(u)$. □

Proof (continuation of Theorem 6). We finish by addressing cases with $m \not\equiv 0 \pmod 5$. Rather than picking between the two instances with equal probability, we skew the distribution. We choose the first of the two instances with a probability denoted as β, to be determined. The expected competitiveness of a deterministic algorithm is $\beta \cdot \frac{3m}{2m+\alpha} + (1 - \beta) \cdot \frac{2m}{2m-\alpha}$. We conclude our proof with a case analysis, choosing β as follows.

$m \bmod 5$	$\lfloor\alpha\rfloor$	$\lceil\alpha\rceil$	β	bound on competitiveness
1	$\frac{2m-2}{5}$	$\frac{2m+3}{5}$	$\frac{2(6m-1)}{5(4m-1)}$	$\frac{20m^2}{16m^2-1}$
2	$\frac{2m-4}{5}$	$\frac{2m+1}{5}$	$\frac{2(12m+1)(3m-1)}{5(24m^2-1)}$	$\frac{30m^2}{24m^2-1}$
3	$\frac{2m-1}{5}$	$\frac{2m+4}{5}$	$\frac{2(12m-1)(3m+1)}{5(24m^2-1)}$	$\frac{30m^2}{24m^2-1}$
4	$\frac{2m-3}{5}$	$\frac{2m+2}{5}$	$\frac{2(6m+1)}{5(4m+1)}$	$\frac{20m^2}{16m^2-1}$

□

Proof (Theorem 7). Consider an arbitrary online, deterministic algorithm. Our adversary begins by presenting a single job with parameter $\langle 0, 5p-1\rangle$. If this were the only job in the instance, the algorithm must accept this job in order to have a bounded competitive ratio. Let t be the time at which the job is started if no other jobs were to arrive. We next introduce a set of m' identical jobs with parameters $\langle t+1, t+2p+2\rangle$, where $m' = m-1$ if $m = 4 \pmod 5$ and $m' = m$ otherwise. Let α denote the number of jobs (including the original job) that are started by the online algorithm on or before time $t+1$. Our adversary will present one of two possible continuations.

In the first, a set of $2m$ new jobs with parameters $\langle t+p+1, t+3p+1\rangle$ are released. OPT achieves all jobs by scheduling the intermediate batch of m' jobs during the region $[t+1, t+p+1)$, the final batch of $2m$ jobs during the region $[t+p+1, t+3p+1)$, and the original job either from $[0, p)$ if $t \geq p-1$, or $[t+3p+1, t+4p+1)$ if $t \leq p-2$. We claim that the online algorithm can achieve at most $2m+\alpha$ jobs, for a competitive ratio of at least $LB1(m,\alpha) = \frac{1+m'+2m}{2m+\alpha}$. With the exception of that first job started by the algorithm at time t, all jobs arrive on or after $t+1$ and have a deadline of at most $t+3p+1$. The algorithm can achieve at most three jobs per machine, and even that requires the machine to start a job on or before time $t+1$. So the algorithm can achieve at most $3\cdot\alpha + 2\cdot(m-\alpha) = 2m+\alpha$ jobs.

Our second instance begins with the same $1+m'$ jobs as does the first. It continues with m new jobs having parameters $\langle t+2, t+p+2\rangle$. OPT achieves all jobs by scheduling the last batch of jobs during the region $[t+2, t+p+2$, the intermediate batch of jobs during the region $[t+p+2, t+2p+2)$, and the original job either from $[0, p)$ if $t \geq p-1$ or else from $[t+2p+2, t+3p+2)$ if $t \leq p-2$. The online algorithm must reject at least α of the final batch of jobs, since the machines started at times t or $t+1$ will still be busy at $t+2$. Therefore, the online algorithm achieve at most $(1+m'+m-\alpha)$ jobs, for a competitive ratio of at least $LB2(m,\alpha) = \frac{1+m'+m}{1+m'+m-\alpha}$ on this instance.

Consider a fixed m. The algorithm can choose the value of α, in which case our adversary can select the worse of the the two lower bounds, leading to an algorithm with competitive ratio at least $r(m,\alpha) = \max(LB1(m,\alpha), LB2(m,\alpha))$. We can therefore bound the competitiveness of any deterministic algorithm with the expression

$$r(m) = \min_{\substack{1\leq\alpha\leq m\\ \alpha\in\mathbb{Z}}} r(m,\alpha)$$

We let α^* denote that α that minimizes $r(m,\alpha)$ if we were to relax the constraint to allow nonintegral α. Since $LB1(m,\alpha)$ is strictly decreasing as α increases and $LB2(m,\alpha)$ is strictly increasing as α increases, the *integral* value of α that minimizes $r(m,\alpha)$ must either be $\lfloor\alpha^*\rfloor$, in which case $r(m) = LB1(m, \lfloor\alpha^*\rfloor)$, or $\lceil\alpha^*\rceil$, in which case $r(m) = LB2(m, \lceil\alpha^*\rceil)$.

We first consider when $m \equiv 4 \pmod 5$ and thus $m' = m - 1$. In this case, the arithmetic simplifies to $LB1(m, \alpha) = \frac{3m}{2m+\alpha}$, $LB2(m, \alpha) = \frac{2m}{2m-\alpha}$, and we have that $\alpha^* = \frac{2m}{5}$. By examination, $r(m)$ is achieved by $\lceil \alpha^* \rceil = \frac{2(m+1)}{5}$, in which case $r(m) = LB2(m, 2(m+1)/5) = \frac{5m}{4m-1}$. When $m \not\equiv 4 \pmod 5$ we have that $m' = m$, $LB1(m, \alpha) = \frac{3m+1}{2m+\alpha}$, and $LB2(m, \alpha) = \frac{2m+1}{2m+1-\alpha}$. In this case, we get that

$$\alpha^* = \frac{(2m+1)(m+1)}{5m+2} = \frac{2m}{5} + \frac{11}{25} + \frac{3}{25(5m+2)}.$$

The sum of the first two terms will always be a nonintegral multiple of $\frac{1}{25}$. The final term is guaranteed to be strictly less than $\frac{1}{25}$ and so it does not affect $\lfloor \alpha^* \rfloor$ or $\lceil \alpha^* \rceil$. By examination, the bounds stated in the theorem are achieved with

$$\begin{aligned}
r(m) &= LB1(m, \lfloor \alpha^* \rfloor) = LB1(m, 2m/5) && \text{for } m \equiv 0 \pmod 5 \\
r(m) &= LB2(m, \lceil \alpha^* \rceil) = LB2(m, (2m+3)/5) && \text{for } m \equiv 1 \pmod 5 \\
r(m) &= LB1(m, \lfloor \alpha^* \rfloor) = LB1(m, (2m+1)/5) && \text{for } m \equiv 2 \pmod 5 \\
r(m) &= LB2(m, \lceil \alpha^* \rceil) = LB2(m, (2m+4)/5) && \text{for } m \equiv 3 \pmod 5
\end{aligned}$$

□

Proof (Theorem 10). The instance begins with 3 jobs: $A = \langle 0, 1\rangle$, $B = \langle 0, 2\rangle$, and $C = \langle 0, 4\rangle$. We assume, w.l.o.g., that job A is assigned to M_1. We have the following cases:

Case 1: B is assigned to M_1: Two jobs $\langle 1, 2\rangle$ arrive. The algorithm cannot accept them both since M_1 is busy with A and B. OPT runs A and B at time 0, the $\langle 1, 2\rangle$ jobs at time 1, and C at time 2.

Case 2: B and C assigned to M_2: A job $\langle 0, 1\rangle$ and four jobs $\langle 2, 4\rangle$ arrive. The algorithm only gets three of the $\langle 2, 4\rangle$ jobs. OPT can run A and the $\langle 1, 2\rangle$ job at time 0, B and C at time 1, and the four other jobs starting at time 2.

Case 3: C assigned to M_1, B assigned to M_2: A job $\langle 0, 1\rangle$ and job $D = \langle 1, 3\rangle$ arrive. To remain competitive, the algorithm must accept both, starting A and the $\langle 0, 1\rangle$ job at time 0. At time 1, jobs B, C, and D remain, with B needing to start immediately. Depending on where job D was assigned, we have one of the following:

Case 3a: D assigned to M_1: A job $\langle 1, 2\rangle$ and two jobs $\langle 3, 4\rangle$ arrive. The algorithm will be unable to achieve all eight jobs. With B on M_2, the $\langle 1, 2\rangle$ must be run on M_1. Yet then C and D will consume M_1 until time 4, and one of the $\langle 3, 4\rangle$ jobs is rejected. OPT can run A and $\langle 0, 1\rangle$ at time 0, B and $\langle 1, 2\rangle$ at time 1, C and D at time 2, and the two $\langle 3, 4\rangle$ jobs at time 3.

Case 3b: D assigned to M_2. Two jobs $\langle 2, 3\rangle$ arrive. The algorithm cannot accept both since M_2 must run B and D by time 3. OPT can run A and $\langle 0, 1\rangle$ at time 0, B and D at time 1, the two $\langle 2, 3\rangle$ jobs at time 2, and C at time 3.

The lowest ratios come from cases 2 and 3a, with the algorithm running at most 7 of the 8 jobs. □