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, dbunde@knox.edu
[2] Dept. of Math. and Computer Science, Saint Louis University, 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.

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 a natural model, for example when distributing incoming requests to a server farm or computer cluster 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 FirstFit algorithm for the immediate dispatch model. In short, it fixes an ordering of the m machines $M_1, \dots, M_m$, 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 deterministic lower bound 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 regarding the deterministic and randomized competitiveness of the models is given in Tables 1 and 2. Due to space limitations, some proofs are omitted from this version of the paper.

m:	1	2	3	4	5	6	7	8	∞
unit-length UB	1 (using EDF)								
equal-length LB	2	1.5	1.4	1.333	**1.333**	1.3	**1.294**	**1.308**	**1.25**
equal-length UB	2	1.5							

Unrestricted or Immediate Notification

unit-length LB		**1.143**							
unit-length UB		**1.333**	**1.421**	**1.463**	**1.487**	**1.504**	**1.515**	**1.523**	**1.582**
equal-length LB		**1.667**	**1.5**	**1.5**	**1.429**	**1.444**	**1.4**	**1.417**	**1.333**
equal-length UB		**1.667**							

Immediate Dispatch

unit-length LB		1.678	1.626	1.607	1.599	1.594	1.591	1.589	1.582
equal-length LB		1.8							
equal-length UB	2	1.8	1.730	1.694	1.672	1.657	1.647	1.639	1.582

Immediate Decision

Table 1. A summary of deterministic lower and upper bounds on the achievable competitiveness for various models. Entries in bold are new results presented in this paper.

m:	1	2	3	4	5	6	7	8	∞
Notification	1.333	**1.263**	**1.256**	**1.255**	**1.25**	**1.252**	**1.251**	**1.251**	**1.25**
Dispatch		**1.333**							
Decision		1.333							

Table 2. A summary of randomized lower bounds for the problem with equal-length jobs. Entries in bold are new results presented in this paper. The only non-trivial upper bound using randomization is a $\frac{5}{3}$-competitive algorithm for the unrestricted model on a single machine [5].

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. Two-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. Goldman et al. [9] show that any *randomized* algorithm can be at best $\frac{4}{3}$-competitive, but 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 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 those objectives, 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. We consider a nonpreemptive model. To complete a job j, the scheduler must commit a machine to it for p consecutive time units during the interval $[r_j, d_j)$. The scheduler's goal is to maximize the number of jobs completed on time. We use competitive analysis, considering the worst-case over all instances of the ratio between the optimal throughput and that produced by an online policy [3, 13, 15]. We presume that an online scheduler has no knowledge of a job request until the job is released. Once released, all of a job's parameters become known to the scheduler[3]. We note the important distinction between having equal-length jobs and *unit-length* jobs. With $p > 1$, the algorithm may start (nonpreemptively) executing one job, and learn of another job that is released while the first is executing. In the unit-length model (i.e., $p = 1$), such a scenario is impossible.

2 The FirstFit Algorithm

We define the FirstFit algorithm 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 (i.e., 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 the minimal M_k for which $Q_k^j(t) \cup \{j\}$ remains feasible, rejecting it if infeasible on all machines. Unlike the BestFit algorithm for the immediate decision model [6],

[3] When jobs share a release time, there are two distinct models. FirstFit operates in an *online-list* model in which those jobs arrive in arbitrary order and the scheduler dispatches or rejects each job before learning of the next. All except the last of our lower bounds apply in the more general *online-time* model, where a scheduler learns about all jobs released at a given time before making decisions about any of them.

FirstFit allows each machine to reorder its queue using the Earliest-Deadline-First (EDF) rule each time it starts running a job from its queue (as an aside, EDF is also used to perform the feasibility test of $Q_k^j(t) \cup \{j\}$ when j is released).

In the remainder of this section, we prove two theorems about FirstFit. In Section 2.1, we show that FirstFit is $\frac{5}{3}$-competitive for equal-length jobs on two machines; this is the best-possible deterministic competitiveness, as later shown in Theorem 5. In Section 2.2 we show, for the special case of unit-length jobs, that FirstFit is $1/\left(1 - \left(\frac{m-1}{m}\right)^m\right)$-competitive for any m.

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 first time $v > u$ 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.

We introduce the following notations. We let $S^{\text{FF}}(t)$ and $S^{\text{Opt}}(t)$ denote the sets of jobs started *strictly* before time t by FirstFit and Opt respectively. 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 $B^{\text{Opt}}(t)$ 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:

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

Intuitively, these functions represent payments for work done in the respective schedules. In the end, we 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 an advanced credit of 2 points for a job in its queue that Opt has already started. The algorithm gets the 3 other points when it starts the delayed job. In contrast, we immediately award Opt its full 3 credits for delayed jobs. We will show that Opt has limited opportunities to carry jobs from one region to the next as delayed; we pay for those discrepancies in advance.

The payment of 2 for jobs in $B^{\text{Opt}}(t)$ is a technical requirement related to our division of time into regions. By definition, each job that FirstFit starts on M_1 completes by the region's end. However, a job started on M_2 may execute past the region's end, possibly hurting it in the next region. We account for this by prepaying Opt during the earlier region for progress made during the overhang.

Lemma 1. *If* FirstFit *rejects job j, all machines are busy during $[r_j, d_j - p)$.*

Proof. If M_k for $k \in \{1, 2\}$ were idle at a time t, its queue is empty. For $t \in [r_j, d_j - p)$, this contradicts j's rejection, as $Q_k^j(r_j) \cup \{j\}$ is feasible by scheduling $Q_k^j(r_j)$ during $[r_j, t)$ as done by the algorithm, and j from $[t, t + p)$. □

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. Consider j with $d_j \geq v + p$ started during $[u, v)$. The set $Q_1^j(r_j) \cup \{j\}$ must be feasible on M_1 at time r_j; this is demonstrated by using the algorithm's actual schedule for $[r_j, v)$, followed by j during $[v, v + p)$, and, based on our definition of v, set $Q_1(v)$ starting at $v + p$ Therefore, such j must have been assigned to M_1 and started at some time $u \leq t \leq v - p$. We note that $Q_1(t)$ 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 $v + p$. If $t > u$, this feasibility of $Q_1(t)$ relative to time $t + p$ contradicts our choice of v (rather than t) as the region's end. Therefore, j must be started on M_1 at time u. □

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

Proof. M_1's idleness implies that $Q_1(u) = Q_1(u + 1) = \emptyset$. Therefore, $v = u + 1$ by definition. Any job started by Opt at time u must have been earlier accepted and completed on M_1 by FirstFit, given its feasibility at a time when M_1 idles. We conclude that $\Phi^{\mathrm{FF}}(v) = \Phi^{\mathrm{FF}}(u)$ and $\Phi^{\mathrm{Opt}}(v) = \Phi^{\mathrm{Opt}}(u)$ □

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

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 $\Phi^{\mathrm{Opt}}(v) - \Phi^{\mathrm{Opt}}(u)$, partitioned as follows.

$3 \cdot$ d due to $d \geq 0$ jobs that are newly added to $D^{\mathrm{Opt}}(v)$. 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 deadline of v+p or later, thus $d \leq 1$.

$3 \cdot$ a due to $a \geq 0$ jobs that are newly added to $S^{\mathrm{Opt}}(v)$, not previously credited as part of $D^{\mathrm{Opt}}(u)$ or $B^{\mathrm{Opt}}(u)$, 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 $S^{\mathrm{FF}}(v)$ or $D^{\mathrm{FF}}(v)$.

$3 \cdot$ r due to $r \geq 0$ jobs that are newly added to $S^{\mathrm{Opt}}(v)$, not previously credited as part of $B^{\mathrm{Opt}}(u)$, 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, Opt must start the jobs denoted by a, r, and b_{old} 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.

$\mathbf{2 \cdot n_2 \geq (r + b_{new})}$
We claim that jobs denoted by r and b_{new} must be started by Opt at times when FirstFit is running one of the jobs denoted by n_2 on M_2, and thus that $r + b_{\text{new}} \leq 2 \cdot n_2$ since Opt may use two machines. Intuitively, this is due to Lemma 1 for jobs of r, and by the definition of $B^{\text{OPT}}(t)$ for jobs of b_{new}. The only technical issue is that if Opt starts a job when M_2 is running a job that started strictly before time u (but overhangs), the job of Opt belongs to $B^{\text{OPT}}(u)$, and thus does not contribute to r or b_{new}.

To complete the proof, we consider $\Phi^{\text{FF}}(v) - \Phi^{\text{FF}}(u)$. By our definitions, this is at least $3(n_1 + n_2) + 2(a + d)$, as jobs for a and d were not credited within $D^{\text{FF}}(u)$. If $n_1 - n_2 \geq d$, these bounds suffice for proving the claim. If $n_1 - n_2 < d$, it must be that $n_1 = n_2$ and $d = 1$. Extra contributions toward Φ^{FF} can be claimed by a further case analysis depending on whether $n_1 = 1$. Details are omitted. □

Theorem 1. FirstFit *is* $\frac{5}{3}$*-competitive for* $m = 2$ *and equal-length jobs.*

Proof. Initially, $\Phi^{\text{OPT}}(0) = \Phi^{\text{FF}}(0) = 0$. Repeated applications of Lemma 3 or 4 for regions $[u, v)$ imply $\Phi^{\text{OPT}}(\infty) \leq \Phi^{\text{FF}}(\infty)$, thus $3 \cdot |S^{\text{OPT}}(\infty)| \leq 5 \cdot |S^{\text{FF}}(\infty)|$. We conclude that $\frac{\text{OPT}}{\text{FF}} \leq \frac{5}{3}$. □

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 $j \in Q_k(t)$ has the largest deadline of jobs in the queue for any $r_j \leq t \leq s_j$. For the sake of contradiction, assume jobs j and j' are in the queue at such time, 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.

Next, 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 $1 \leq k \leq m$. Note that $\sum_{k=1}^{m} Y_k = m^m - (m-1)^m$ is a geometric sum with 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 since 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 $\sum_{z=1}^{m} Y_z = m^m - (m-1)^m$, thus each job j run by FirstFit collects at most m^m units of charge. □

Our analysis of FirstFit is tight. Consider $m + 1$ "waves" of jobs. For $1 \leq w \leq m$, wave w has $m \cdot Y_{m+1-w}$ jobs released at $\sum_{z=1}^{w-1} Y_{m+1-z}$ with deadline m^m. The last wave has $m \cdot (m - 1)^m$ jobs released at time $m^m - (m - 1)^m$ with deadline m^m. FirstFit dispatches wave i to machine M_i, using it until time m^m. FirstFit must reject the last $m(m - 1)^m$ jobs and runs only $m \cdot \sum_{k=1}^{m} Y_k = m(m^m - (m - 1)^m)$ jobs. Opt runs all $m \cdot m^m$ jobs by distributing each wave across all m machines, giving 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 $\langle r_j, d_j \rangle$ to denote a job with release time r_j and deadline d_j. Goldman et al. provide a $\frac{4}{3}$-competitive lower bound for randomized algorithms on one machine in the unrestricted model [9]. Their construction does not apply to multiple machines in the unrestricted model, but Ding et al. use such a construction in the *immediate decision* model to provide a randomized lower bound of $\frac{4}{3}$ for any m [6]. We first show that this bound applies to the *immediate dispatch* model.

Theorem 3. *For the* immediate dispatch *model with $p \geq 2$, every randomized algorithm has a competitive ratio at least $\frac{4}{3}$ against an oblivious adversary.*

Proof. We apply Yao's principle [3], bounding the expected performance of a *deterministic* algorithm against a *random* distribution. In particular, we consider two possible instances, both beginning with m jobs denoted by $\langle 0, 2p + 1 \rangle$. For a given deterministic algorithm, let α be the number of machines, at time 0, that start a job or have two jobs already assigned. Our first instance continues with m jobs having parameters $\langle p, 2p \rangle$. The $m - \alpha$ machines that were assigned less than two jobs and that idle at time 0 can run at most one job each. The other α machines run at most 2 jobs each. Overall, the algorithm runs at most $2 \cdot \alpha + (m - \alpha) = m + \alpha$ jobs, for a competitive ratio of at least $\frac{2m}{m+\alpha}$. Our second instance continues with m jobs having parameters $\langle 1, p + 1 \rangle$. At least α of these are rejected, since none can run on the α machines that are otherwise committed, making the competitive ratio at least $\frac{2m}{2m-\alpha}$. On a uniform distribution over these two instances, the deterministic algorithm has an expected competitive ratio of at least $\frac{1}{2}\left(\frac{2m}{m+\alpha} + \frac{2m}{2m-\alpha}\right)$, which is minimized at $\frac{4}{3}$ when $\alpha = \frac{m}{2}$. □

We can prove slightly stronger bounds for *deterministic* algorithms, since an adversary can apply the worse of two instances (rather than their average).

Theorem 4. *For the* immediate dispatch *model with $p \geq 2$ and m odd, no deterministic algorithm has a competitive ratio strictly better than $\frac{4m}{3m-1}$.*

Proof (sketch). For the same two instances as in Theorem 3, $\max(\frac{2m}{m+\alpha}, \frac{2m}{m-\alpha})$ is minimized at $\frac{4m}{3m-1}$ with $\alpha = \lfloor \frac{m}{2} \rfloor = \frac{m-1}{2}$ or $\alpha = \lceil \frac{m}{2} \rceil = \frac{m+1}{2}$. □

Theorem 5. *For the* immediate dispatch *model with $p \geq 3$ and m even, no deterministic algorithm has a competitive ratio strictly better than $\frac{4m+2}{3m}$.*

Proof (sketch). We adapt our construction, starting with one job $\langle 0, 4p+1\rangle$, which we assume is started at time t by the algorithm. We next release m jobs $\langle t+1, t+2p+2\rangle$, and let α denote the number of machines at time $t+1$ that are either running a job or have two jobs already assigned. We release a final set of m jobs, either all $\langle t+p+1, t+2p+1\rangle$ or all $\langle t+2, t+p+2\rangle$. In the first case, an algorithm gets at most $m+\alpha$, and in the second at most $1+2m-\alpha$. The lower bound of $\max(\frac{1+2m}{m+\alpha}, \frac{1+2m}{1+2m-\alpha})$ is minimized at $\frac{4m+2}{3m}$ when $\alpha = \lfloor \frac{1+m}{2} \rfloor = \frac{m}{2}$. $\square$

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 $\frac{m}{3}$ 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. We do this by doubling the size of the second wave in one of the two instances, thereby changing the balancing point of the optimal behavior for the construction.

Theorem 6. *For the* unrestricted *model with $p \geq 2$, no randomized algorithm has a competitive ratio strictly better than the following, with m given mod 5:*

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

Proof (sketch). We use Yao's principle with a distribution of two instances. Both instances begin with m jobs $\langle 0, 2p+1\rangle$. For a fixed deterministic algorithm, let α be the number of machines that start a job at time 0. Our first instance continues with $2m$ jobs $\langle p, 3p\rangle$. A machine that is not starting a job at time 0 can run at most 2 jobs. Therefore, an online algorithm completes at most $3\alpha + 2\cdot(m-\alpha) =$

$2m+\alpha$ jobs, for a competitive ratio of at least $\frac{3m}{2m+\alpha}$ on this instance. Our second instance continues with m jobs $\langle 1, p+1\rangle$. An 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 least $\frac{2m}{2m-\alpha}$ 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 least $\frac{1}{2}\left(\frac{3m}{2m+\alpha}+\frac{2m}{2m-\alpha}\right)$, minimized at $\frac{5}{4}$ when $\alpha=\frac{2m}{5}$. This completes the theorem for $m \equiv 0 \pmod 5$. For other modularities of m, an even stronger bound holds because the algorithm cannot choose $\alpha = \frac{2m}{5}$. □

Our next theorem strengthens the bound for deterministic algorithms by first releasing a single job with large deadline (similar to Theorem 5).

Theorem 7. *For the* unrestricted *model with $p \geq 3$, no deterministic algorithm has a competitive ratio strictly better than the following, with m given mod 5:*

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

Proof (sketch). We release a job $\langle 0, 5p-1\rangle$, which we assume is started at time t by the algorithm. Next, we release m' jobs $\langle t+1, t+2p+2\rangle$, where $m' = m-1$ if $m = 4 \pmod 5$ and $m' = m$ otherwise. Let α be the number of jobs (including the first) started on or before time $t+1$. Our adversary continues in one of two ways, releasing either $2m$ jobs with parameters $\langle t+p+1, t+3p+1\rangle$ or m jobs with parameters $\langle t+2, t+p+2\rangle$. These choices give competitive ratios of at least $\frac{1+m'+2m}{2m+\alpha}$ and $\frac{1+m'+m}{1+m'+m-\alpha}$ respectively. The precise lower bounds come from optimizing α for varying values of m. □

The construction of Theorem 7 requires $p \geq 3$, to leverage the introduction of the job $\langle 0, 5p-1\rangle$. For $p = 2$, the following bound can be shown using the construction from Theorem 6, and deterministic choice $\alpha = \lfloor\frac{2m}{5}\rfloor$ or $\alpha = \lceil\frac{2m}{5}\rceil$.

Theorem 8. *For the* unrestricted *model with $p = 2$, no deterministic algorithm has a competitive ratio strictly better than the following:*

$m \equiv 0$	$m \equiv 1$	$m \equiv 2$	$m \equiv 3$	$m \equiv 4$
$\frac{5}{4}$	$\frac{15m}{12m-2}$	$\frac{10m}{8m-1}$	$\frac{15m}{12m-1}$	$\frac{5m}{4m-1}$

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 to $p = 1$; an adversary cannot force the rejection of new jobs due to machines that are committed to other tasks. With the following theorems, we provide (weaker) lower bounds for unit-length jobs, 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 be better than 9/8-competitive. A deterministic online-list algorithm cannot be better than 8/7-competitive.*

4 Conclusions

In this paper, we have introduced a study of the *immediate dispatch* model when 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. The primary open problem is to develop stronger algorithms for $m \geq 3$ in any of these models.

Acknowledgments We thank the referees for helpful comments. D.P. Bunde was supported in part by Howard Hughes Medical Institute grant 52005130.

References

1. Avrahami, N., Azar, Y.: Minimizing total flow time and total completion time with immediate dispatching. Algorithmica **47**(3), 253–268 (2007)
2. Baruah, S.K., Haritsa, J.R., Sharma, N.: On-line scheduling to maximize task completions. J. Combin. Math. and Combin. Computing **39**, 65–78 (2001)
3. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, New York (1998)
4. Chin, F.Y.L., Chrobak, M., Fung, S.P.Y., Jawor, W., Sgall, J., Tichý, T.: Online competitive algorithms for maximizing weighted throughput of unit jobs. J. Discrete Algorithms **4**(2), 255–276 (2006)
5. Chrobak, M., Jawor, W., Sgall, J., Tichý, T.: Online scheduling of equal-length jobs: Randomization and restarts help. SIAM Journal on Computing **36**(6), 1709–1728 (2007)
6. Ding, J., Ebenlendr, T., Sgall, J., Zhang, G.: Online scheduling of equal-length jobs on parallel machines. In Arge, L., Hoffmann, M. (eds.) ESA 2007. LNCS, vol. 4698, pp. 427–438. Springer, Heidelberg (2007)
7. Ding, J., Zhang, G.: Online scheduling with hard deadlines on parallel machines. In Cheng, S.W., Poon, C.K. (eds.) AAIM 2006. LNCS, vol. 4041, pp. 32–42. Springer, Heidelberg (2006)
8. Ebenlendr, T., Sgall, J.: A lower bound for scheduling of unit jobs with immediate decision on parallel machines. In Bampis, E., Skutella, M. (eds.) WAOA 2008. LNCS, vol. 5426, pp. 43–52 (2008)
9. Goldman, S., Parwatikar, J., Suri, S.: On-line scheduling with hard deadlines. J. Algorithms **34**(2), 370–389 (2000)
10. Goldwasser, M.H., Kerbikov, B.: Admission control with immediate notification. J. Scheduling **6**(3), 269–285 (2003)
11. Goldwasser, M.H., Misra, A.B.: A simpler competitive analysis for scheduling equal-length jobs on one machine with restarts. Information Processing Letters **107**(6), 240–245 (2008)
12. Goldwasser, M.H., Pedigo, M.: Online nonpreemptive scheduling of equal-length jobs on two identical machines. ACM Trans. on Algorithms **5**(1), Article 2, 18 pages (November 2008)
13. Karlin, A., Manasse, M., Rudolph, L., Sleator, D.: Competitive snoopy paging. Algorithmica **3**(1), 70–119 (1988)
14. Pruhs, K.: Competitive online scheduling for server systems. SIGMETRICS Perform. Eval. Rev. **34**(4), 52–58 (2007)
15. Sleator, D., Tarjan, R.: Amortized efficiency of list update and paging rules. Communications of the ACM **28**, 202–208 (1985)