# Online Scheduling of Unit Length Jobs with Commitment and Penalties

Stanley Fung



# Online Scheduling of Unit Length Jobs with Commitment and Penalties

Stanley P. Y. Fung

Department of Computer Science, University of Leicester, Leicester, United Kingdom.
pyf1@le.ac.uk

**Abstract.** We consider the online scheduling of unit length jobs with two models of commitment. In immediate notification, the acceptance of a job must be decided as soon as it is released. In immediate decision, the actual time slot allocated to the job must also be fixed at the job's arrival as well. Failure to honour a commitment will result in a penalty. The non-commitment version has been extensively studied. In this paper we give algorithms and lower bounds for the two models of commitment. For immediate decision, we give an $O(m(1+\rho)^{1/m})$-competitive algorithm where $m$ is the number of machines and $\rho$ is the penalty factor, and when $m$ is large we give an $O(\log(1+\rho))$ upper bound. This is matched by a lower bound of $\Omega(\log \rho)$ on the competitive ratio. For immediate notification we give a lower bound of $\Omega(\log \rho / \log\log \rho)$. We also give some better bounds when $m = 1$ or when $\rho$ is small. Finally we give considerations to the case of separate arrival and start times.

## 1 Introduction

*The model.* The basic setting of our problem is as follows: there is a set of jobs, where each job $j$ is specified by a triple $(r(j), d(j), w(j))$ representing its release time, deadline and weight respectively. Each job has length 1 and all release times and deadlines are integers; also, time is discrete and is divided into time slots of unit length. A *slot* is defined as a tuple $(i, t)$ which is a combination of a machine $i$ and a time step $[t, t+1)$. Thus, each job will fit into a slot and there will be no interruption while a job is being executed.

Jobs arrive online, so a job is only known to the algorithm upon its release time, at which point all information about the job is known. All jobs arriving at the same time step is known to the algorithm at the same time. The objective of the algorithm is to schedule these jobs on $m$ identical machines so as to maximize the profit, i.e. the total weight of jobs completed before their deadlines. This is known as the *unit job scheduling* problem in the literature.

As the algorithms are online, they do not have knowledge of future information and have to make scheduling decisions based on what is known. This inevitably leads to suboptimal algorithms, and the standard way to analyze such online algorithms is *competitive analysis*. An algorithm (for a maximization problem) is *$R$-competitive* if the value returned by the algorithm is always at least $1/R$ times the value returned by an offline optimal algorithm, for any

input instance. In this paper we will use $\mathcal{OPT}$ to denote the offline optimal algorithm, and $\mathcal{ONL}$ to denote the online algorithm in question. Also, we use $\mathcal{OPT}(i,t)$ to denote the job scheduled by $\mathcal{OPT}$ at slot $(i,t)$; similar definition holds for $\mathcal{ONL}(i,t)$.

In the above setting, whether a job will be completed on time is only known until its deadline is reached. The algorithm is not required to announce upfront whether it intends to complete the job. In many application areas where customer service is crucial, it is important to let the customers (users supplying the jobs) know as early as possible whether (or when) the job will be completed. Ideally this should be made known immediately, at the same time the job is presented; if the job is rejected the customer can then find another company to fulfill the job, and not be left in a situation where he/she only finds out that the job is not completed at the last minute (and therefore is too late to do anything about it). Such 'commitment' in job scheduling can be modelled in a number of ways, two of which we consider in this paper:

**Immediate notification (IMM NOTIF):** when a job is released, the algorithm has to decide immediately whether to accept the job. An accepted job must be completed before its deadline. A job not accepted is gone forever. This model was introduced in [14].

**Immediate decision (IMM DEC):** In addition to IMM NOTIF, here the exact time to execute an accepted job, and also the machine to execute the job (if there are more than one machine) must also be announced at the same time that the job is accepted, and this cannot be changed later. This model was introduced in [9].

While it would be ideal to always keep to one's commitments, unfortunately it is not uncommon for (for example) online shopping or delivery companies to delay or miss orders altogether. In our model, an accepted job can be *evicted*[1] later, but such eviction incurs a penalty that is proportional to the weight of the job evicted. This is in addition to losing the weight of the job. A job once evicted is gone and will not come back. In the case of IMM DEC, reallocating an accepted job to a different slot is not allowed, even after paying penalties, so the job is gone forever. The objective of the algorithm is then to maximize the total weight of all completed jobs, minus the sum of all penalties incurred. Observe that $\mathcal{OPT}$ does not pay penalties as clearly it can decide in advance whether to accept a job and when to schedule it, and to announce it upfront, without damaging its profit.

We use $\rho$ to denote the *penalty factor*, where $\rho \geq 0$, and the penalty of evicting a job $j$ of weight $w(j)$ is then equal to $\rho w(j)$. There are several motivations of studying such a penalty model. Incorporating a penalty factor allows us to study quantitatively the cost of breaking a commitment. It may be tempting to suggest that $\rho$ should be at most 1, so that the penalty is at most the value of the job, and

[1] We use the term 'eviction' rather than 'preemption' to distinguish with the case where a job is interrupted in the middle of its execution (which will not happen in our setting.)

it may well be true in practice. While we do study small $\rho$ values, we also allow $\rho$ to grow arbitrarily large and then study how the competitive ratio depends on $\rho$; this is because when $\rho \to \infty$ it models the situation when eviction is not allowed, which was the case in earlier works. A finite but large $\rho$ models an intermediate situation where it is very costly but not impossible to break a commitment. This penalty model also allows the study of scheduling weighted jobs, to model the different importance of jobs; in nonpreemptive models like [8] allowing weighted jobs would lead to trivial bad results.

The idea of giving penalties for not processing jobs is not new; for example there are a lot of work on scheduling with rejection (see e.g. [19] for a survey). Also, there are many works in management science and operations research concerning *quoted lead time*, i.e. it is the supplier, on receiving a demand, who quote a time when the order will be met, and typically the profit received is a non-increasing function of this quoted lead time; see e.g. [16]. Most of these were not studied with competitive analysis, but [17] studied a model where the profit decreases by a fixed amount per unit of added time (this is the penalty) but commitment is not breakable (called 100% reliable).

The need of immediate commitment and revocation penalties can also be found in mechanism design in auctions. In [6] an auction problem for unit sized time slots is considered. The auction must be truthful, i.e. players are incentivized to bid their true values. Existing algorithms for unit job scheduling [18] already satisfy this property, but [6] considered an additional requirement of *promptness*, which can be guaranteed by restricting the algorithm to allocate a job to a fixed time slot on its arrival and not change afterwards (although the job can be removed later); it is therefore identical to the immediate decision problem (although there is no penalty so $\rho = 0$.)

Additionally, banner advertisement auctions were studied in [1] and [7] where request acceptance decisions must be made immediately but it is possible to 'buy back' or 'bump' an accepted request by paying a penalty; the papers explained the motivations for the need for immediate commitment and penalties. In Section 4 we discuss in more detail how this relates to our problem with a separation of arrival and starting times.

*Previous results.* The online scheduling of unit length jobs (without any commitment or penalty) has been a very active area of scheduling research in the past ten years or so. It originated from a problem in buffer management [18] but it soon attracted a lot more attention. We refer readers to the survey [13] for a comprehensive literature review of the area. Here we list only the latest results. For a single machine, the current best deterministic algorithm is 1.828-competitive [11], and the lower bound is 1.618 [15, 5, 20]. The randomized upper and lower bounds are respectively 1.582 and 1.25 [4, 5]. For multiple machines the upper and lower bounds are $(1 - (\frac{m}{m+1})^m)^{-1}$ and 1.25 respectively [4].

This unit job scheduling problem with IMM DEC or IMM NOTIF has not been studied before, but there are other existing work on scheduling with commitment (but not penalty), where a committed job must always be completed. Goldwasser and Kerbikov [14] considered IMM NOTIF where jobs have unequal

lengths and the objective is to maximize the total length of completed jobs. They showed that in many cases there are algorithms with competitive ratios that are not worse than their non-commitment counterparts. Ding et al. [8] considered IMM DEC with equal length (but non-unit length) and unweighted jobs (i.e. the objective is to maximize the number of completed jobs), and gave an algorithm with a competitive ratio that tends to $e/(e-1) \approx 1.58$ when $m$ is large. A $e/(e-1)$ lower bound (again for large $m$) from [10] shows that this algorithm is optimal; in fact the lower bound holds even for unit length jobs. In [2] an *immediate dispatch* model was proposed, in which the machine allocated for an accepted job must be announced upfront but the machine is free to move these jobs around. In [3] an online scheduling problem with commitment (IMM NOTIF) was studied where the penalty is equal to the length of the unfinished part of the job, with application in the charging of electric vehicles.

Allowing a penalty factor with commitment was considered in [12], where it studied longer (non-unit length) jobs and also addressed a related problem of *preemption penalty*. They considered two cases where (i) jobs have different lengths but their values are proportional to their lengths, or (ii) jobs have equal lengths but arbitrary weights. Several algorithms with competitive ratios depending on the penalty factor $\rho$ were given. As already mentioned, commitment with penalty also arises in auctions; [1] and [7] gave algorithms but the items to be auctioned do not spread over a time period (even though the requests do.)

*Our results.* In this paper we consider a setting similar to [12] but with unit length jobs. This may look like a simpler problem, but it allows us to remove any consideration of interruption due to job arrival in the middle of job execution and thus study purely the effect of commitment on the scheduling algorithms' performance. As we will shortly see, this allows sublinear (in $\rho$) competitive ratios which have not yet been achieved in the longer job case.

For IMM DEC we give an upper bound of $O(m(1+\rho)^{1/m})$, and a lower bound of $\Omega(\log \rho)$ (that applies to any number of machines). Note that when $m = \Theta(\log \rho)$ the bounds asymptotically match. When $m > \log \rho$ we modify the algorithm so that the competitive ratio remains $O(\log \rho)$. In the case of one machine we give $\Theta(\rho)$ bounds. For IMM NOTIF, we give an $\Omega(\log \rho / \log\log \rho)$ lower bound on the competitive ratio for any number of machines. For small values of $\rho$ and for a single machine, we give another lower bound. Table 1 summarizes the results. Finally we tighten the bound for IMM NOTIF if arrival times and starting times are separate. Because of space constraints some proofs are omitted.

## 2 Immediate Decision

### 2.1 Upper Bound

The algorithm maintains a *provisional schedule* $S$ to record which slot a pending job is scheduled to be executed. Each time a job is accepted, it will be provisionally assigned a slot in $S$. For each machine $i$ and time step $t$ on or after the

| $m$ | 1 | | | | | | | | large |
|---|---|---|---|---|---|---|---|---|---|
| $\rho$ | 0 | 0.1 | 0.2 | 0.5 | 1 | 1.5 | 2 | large | large |
| IMM DEC UB | 2 | 3.727 | 4.760 | 6 | 8 | 10 | 12 | $O(\rho)$ | $O(\log \rho)$ |
| IMM DEC LB | 2 | 2 | 2 | 2 | 2 | 2.5 | 3 | $\Omega(\rho)$ | $\Omega(\log \rho)$ |
| IMM NOTIF UB | 2 | 3.117 | 3.727 | 5.236 | 7.464 | 9.583 | 11.657 | $O(\rho)$ | $O(\log \rho)$ |
| IMM NOTIF LB | 1.618 | 1.691 | 1.766 | 2 | 2 | 2 | 2 | $\Omega(\log \rho / \log\log \rho)$ | |

**Table 1.** Some typical values of the upper bounds (UB) and lower bounds (LB) on the competitive ratios.

current time, $S(i,t)$ denotes the job that is provisionally assigned to slot $(i,t)$ in $S$. If no job is assigned to a slot, we imagine that a null job with weight 0 is assigned there.

A simple algorithm would be to accept a job $j$ if it is 'sufficiently heavy':

**Algorithm 1.** Let $\beta = 2(1+\rho)$ if $\rho > (\sqrt{2}-1)/2$, and $\beta = 1+\rho+\sqrt{\rho^2+\rho}$ otherwise. At every time step $t$, the algorithm runs an admission procedure to update $S$, and then simply executes the jobs $S(i,t)$, $1 \le i \le m$. The admission procedure considers each new job $j$ arriving in that time step in turn (the order in which they are considered does not matter). It finds the slot $(i,u)$ before $d(j)$ with the minimum $w(S(i,u))$. Let $k$ be the job in this slot, i.e., $k = S(i,u)$. If $w(j) > \beta w(k)$, accept $j$ in slot $(i,u)$ and $k$ is *evicted*. As this is IMM DEC, $k$ is lost. Otherwise $j$ is rejected.

**Theorem 1.** *Algorithm 1 is* $\min(4(1+\rho), 4\rho+2+4\sqrt{\rho^2+\rho})$*-competitive for any* $m$*.*

The competitive ratio of Algorithm 1 grows linearly with $\rho$. In fact we can give a better algorithm if $m > 1$. The key idea is to make sure that for all jobs committed to the same time step, their weights should differ 'substantially.'

**Algorithm 2.** Let $\beta = (2\rho+2)^{1/m}$. For each $t$, let $S'(i,t), 1 \le i \le m$, be the jobs of $S(i,t)$ sorted in decreasing order of job weights, i.e., $S'(i,t)$ denotes the $i$-th largest-weight job scheduled to run at time step $t$ in the current provisional schedule. (As already noted, if there are fewer than $m$ such jobs, we assume the provisional schedule is filled with weight-0 jobs.) The algorithm maintains the property that $w(S'(i,t)) \ge \beta w(S'(i+1,t))$ for all $i$ and $t$.

Only the admission procedure is changed from Algorithm 1. For each new job $j$ arriving at this time step $t$ (again the order is not important), the algorithm tries to find a time $u \ge t$ such that $u < d(j)$ and $w(j) \ge \beta w(S'(1,u))$, i.e., $j$ is heavy enough relative to the heaviest job committed to time $u$. If there are more than one such $u$, choose anyone. Evict the lightest job at that time step, $S'(m,u)$, replacing it with $j$. The job $S'(m,u)$ is called the job *evicted* by $j$. As this is IMM DEC, this evicted job is lost forever and will not be considered again. If no such $u$ exists, reject $j$. Note that after the eviction, $j$ becomes the new $S'(1,u)$, the old $S'(1,u)$ becomes the new $S'(2,u)$ and so on.

This is only for notational convenience; the jobs stay at the same machines and do not actually move. The property $w(S'(i,u)) \geq \beta w(S'(i+1,u))$ is maintained.

**Theorem 2.** *Algorithm 2 is $O(m(1+\rho)^{1/m})$-competitive.*

*Proof.* For brevity, let $x_i(t) = w(\mathcal{OPT}(i,t))$ and $y_i(t) = w(S'(i,t))$ at the end of the execution. Note that at the end of the execution, $S$ is actually the final schedule and so $y_i(t)$ represents $w(\mathcal{ONL}(i,t))$ but in sorted order of weights.

We map each job $\mathcal{OPT}(i,t)$ to a time step of $\mathcal{ONL}$. If $\mathcal{OPT}(i,t)$ is completed in $\mathcal{ONL}$ as well, we map it to the time step at which it appears in $\mathcal{ONL}$. Otherwise, it is either rejected immediately or is evicted later on. If it is rejected on its arrival, then at the time of its arrival, $S'(1,t)$ must be a job with weight larger than $x_i(t)/\beta$. Furthermore, observe that for any slot, the weight of the job in it can only increase during the course of execution of the algorithm. Thus, $y_1(t)$ must also be at least $x_i(t)/\beta$. We map $\mathcal{OPT}(i,t)$ to time step $t$. Finally, if $\mathcal{OPT}(i,t)$ is evicted in $\mathcal{ONL}$, we associate it with the job that evicts it. If this job is in turn evicted later, the association is transferred to the new evicting job. Thus each evicted job is associated with, possibly via a chain of evictions, a job completed by $\mathcal{ONL}$. Job $\mathcal{OPT}(i,t)$ is then mapped to the time step where it is executed in $\mathcal{ONL}$.

Consider a time step $t$. We bound the total weight of jobs mapped to $t$. All the $\mathcal{OPT}(i,t)$ may be rejected by $\mathcal{ONL}$ and thus mapped to $t$. All the $\mathcal{ONL}(i,t)$ may be completed by $\mathcal{OPT}$ and will be mapped here. Moreover all jobs transitively evicted by $\mathcal{ONL}(i,t)$ may also be mapped here. Note that whenever a job $j_1$ evicts another job $j_2$, it must be that $w(j_1) \geq \beta^m w(j_2)$ because this $j_2$ must be the lightest job $S'(m,u)$ in its time step $u$ and $w(j_1) \geq \beta w(S'(1,u)) \geq \ldots \geq \beta^m w(S'(m,u))$. Thus, the total weight of evicted jobs associated with a job of weight $y_i(t)$ is at most $y_i(t)/\beta^m + y_i(t)/\beta^{2m} + \cdots < y_i(t)/(\beta^m - 1)$. These jobs are all mapped to $t$. No other jobs are mapped to $t$. At time $t$, $\mathcal{ONL}$ gets a profit of $\sum_{i=1}^m y_i(t)$, but as each of the jobs may have evicted a chain of other jobs, there is a penalty of at most $\rho y_i(t)(1/\beta^m + 1/\beta^{2m} + \ldots) < \rho y_i(t)/(\beta^m - 1)$ associated with a job with weight $y_i(t)$. The ratio of mapped $\mathcal{OPT}$ job weights to $\mathcal{ONL}$ job weights, minus penalties, in this time step is therefore at most

$$\frac{\sum_{i=1}^m x_i(t) + \sum_{i=1}^m y_i(t) + \sum_{i=1}^m \frac{y_i(t)}{\beta^m - 1}}{\sum_{i=1}^m y_i(t) - \sum_{i=1}^m \frac{y_i(t)\rho}{\beta^m - 1}} \leq \frac{m\beta y_1(t) + \sum_{i=1}^m y_i(t)\frac{\beta^m}{\beta^m - 1}}{\sum_{i=1}^m y_i(t)(1 - \frac{\rho}{\beta^m - 1})}$$

$$\leq \frac{m(\beta(\beta^m - 1) + \beta^m)y_1(t)}{y_1(t)(\beta^m - 1 - \rho)} = \frac{m((2\rho+2)^{1/m}(2\rho+1) + 2\rho + 2)}{\rho + 1}$$

$$< 2m(2\rho+2)^{1/m} + 2m$$

where we used the fact that $0 \leq y_i(t) \leq y_1(t)$. □

Observe that the competitive ratio $O(m(1+\rho)^{1/m})$ decreases with increasing $m$, from $O(\rho)$ when $m = 1$, to $O(\log(1+\rho))$ when $m = \ln(2+2\rho)$ (noting that $x^{1/\ln x} = e$ for any $x$). But the competitive ratio then increases with increasing $m$.

This is due to the fact that the algorithm effectively uses only one slot 'seriously' in each time step. In fact, for example, if $m$ jobs of the same weight and of tight deadline[2] arrives, the algorithm will only schedule one of them, which clearly makes it not better than $m$-competitive.

For large $m$, we can modify the algorithm to retain the $O(\log(1+\rho))$ competitive ratio. For convenience, assume $m' = \ln(2+2\rho)$ is an integer and that $m$ is a multiple of $m'$. We partition the $m$ machines into $m/m'$ bands, each with $m'$ machines. Each band then effectively runs the algorithm indpendently. Within each band, the machines maintain a provisional schedule with the property that $w(S'(i,t)) \geq \beta w(S'(i+1,t))$ for all $i$ and $t$, where $\beta = (2\rho+2)^{1/m'}$. When a new job $j$ arrives, it tries to get accepted by testing if $w(j) \geq \beta w(S'(1,u))$ for some $u$ and some band, and evict $S'(m',u)$ if so. Job $j$ is only rejected if it cannot be accepted in any band. (If more than one band can accept the job, choose anyone.)

**Theorem 3.** *The modified Algorithm 2 is $O(\log(1+\rho))$-competitive.*

*Proof.* Suppose the bands are numbered 1, 2, ... and $y_i^z(t)$ denotes $w(S'(i,t))$ for band $z$. The scheme that we use to map jobs in $\mathcal{OPT}$ to time steps in $\mathcal{ONL}$ remain the same as in Theorem 2. In this case, if the job $OPT(i,t)$ is rejected on arrival, then for each band, $S'(1,t)$ must contain a job of weight larger than $x_i(t)/\beta$. We group every $m'$ such jobs and associate them with a $y_1^z(t)$ of a band. Also, if a job $j_1$ evicts another job $j_2$ then $w(j_1) \geq \beta^{m'} w(j_2)$. Similar to Theorem 2 the competitive ratio is then

$$\begin{aligned}
&\frac{\sum_{i=1}^{m} x_i(t) + \sum_z \sum_{i=1}^{m'} y_i^z(t) + \sum_z \sum_{i=1}^{m'} y_i^z(t) \frac{\beta^{m'}}{\beta^{m'}-1}}{\sum_z \sum_{i=1}^{m'} y_i^z(t)(1 - \frac{\rho}{\beta^{m'}-1})} \\
&\leq \frac{m' \sum_z \beta y_1^z(t) + \sum_z m' y_1^z(t) \frac{\beta^{m'}}{\beta^{m'}-1}}{\sum_z y_1^z(t)(1 - \frac{\rho}{\beta^{m'}-1})} = \frac{m'(\beta(\beta^{m'}-1) + \beta^{m'})}{(\beta^{m'} - 1 - \rho)} \\
&= \frac{m'((2\rho+2)^{1/m'}(2\rho+1) + 2\rho + 2)}{\rho+1} < m'(2(2\rho+2)^{1/m'} + 2)
\end{aligned}$$

which is $O(\log(1+\rho))$ as $(2\rho+2)^{1/m'} = (2\rho+2)^{1/\ln(2\rho+2)} = e$. □

### 2.2 Lower Bounds

Next we prove a lower bound of $\Omega(\log \rho)$ for any number of machines. The bound only works for large $\rho$. For large number of machines ($m = \Omega(\log \rho)$) it follows from Theorem 3 that the bound is optimal.

**Theorem 4.** *Any deterministic algorithm has competitive ratio $\Omega(\log \rho)$ for the immediate decision model with $m$ machines.*

[2] We call a job $j$ tight if $d(j) = r(j) + 1$, i.e. it must be scheduled immediately.

*Proof.* Define a sequence of integers $D_i$ as follows: $D_1 = D$ for a large $D$, and $D_i = 2(D_{i+1} + 1)$ for $i > 1$. The $D$ is chosen in such a way that $D_1, D_2, \ldots, D_j$ are all integers for some sufficiently large $j$. Note that if the $D_i$'s are large then all $D_i/D_{i+1}$ are very close to (but above) 2. Moreover, observe that $2 + D_{i+1} = 1 + D_i/2$ and hence $(2 + D_{i+1})2^{i-1} = (1 + D_i/2)2^{i-1} = (2 + D_i)2^{i-2} = \ldots = (2 + D_2)2^0 = 1 + D_1/2$ for all $i$.

Suppose $\mathcal{ONL}$ is $R$-competitive. The adversary construction consists of a large number of rounds. In the construction, the minimum job weight is 1 and we will make sure the maximum job weight $W$ is less than $\rho$, so that there is no point in evicting a job to accept another job because even if we evict a minimum weight job to accept a maximum weight one, this will still result in reduced total profit of the schedule. Then without loss of generality we can assume that no eviction takes place.

Let $I_1 = [0, 1 + D_1)$. Round 1 begins at time $t_1 = 0$, when $m(1 + D_1)$ jobs of deadline $1 + D_1$ and weight 1 arrives. Suppose $\mathcal{ONL}$ accepts $x_1$ of these jobs. As $\mathcal{OPT}$ can accept all jobs, for $\mathcal{ONL}$ to be $R$-competitive it must be that $x_1 \geq m(1 + D_1)/R$. At most $m$ of these accepted jobs can be scheduled in $[0, 1)$. Each of the remaining $x_1 - m$ accepted jobs is committed to either $[1, 2 + D_2)$ or $[2 + D_2, 1 + D_1)$, both of length $1 + D_2$. One of them is the denser interval, i.e. the one that contains at least $(x_1 - m)/2$ accepted jobs. Let $I_2 = [t_2, t_2 + D_2 + 1)$ be this denser interval, where $t_2 \in \{1, 2 + D_2\}$. As $\mathcal{ONL}$ obeys immediate decision, the location of $I_2$ is known.

Then Round 2 begins at $t_2$, when $m(D_2 + 1)$ jobs of deadline $t_2 + D_2 + 1$ and weight 2 arrives. Clearly they can only be scheduled in $I_2$. Suppose $\mathcal{ONL}$ accepts $x_2$ of these jobs. $\mathcal{OPT}$ can always fill slots in $I_2$ with weight-2 jobs, and fill slots in $I_1 - I_2$ with weight-1 jobs. Thus, $\mathcal{OPT}$ can get a profit of $m(2+D_2)+2m(1+D_2)$. So in order for $\mathcal{ONL}$ to be $R$-competitive, it must be that $x_1 + 2x_2 \geq m(4 + 3D_2)/R$. There are now at least $(x_1 - m)/2 + x_2 - m$ accepted jobs in the interval $[t_2+1, t_2+D_2+1)$, and we can again divide it into two halves, with one of them being denser, i.e. containing at least half of the accepted jobs. Let $I_3 = [t_3, t_3 + D_3 + 1)$ be this denser interval, with $t_3 \in \{t_2+1, t_2+D_3+2\}$. In this denser interval, there are at least $(x_1 - m)/4 + (x_2 - m)/2$ accepted jobs. We then proceed to the next round where jobs will arrive into this denser interval.

In general, just after round $i-1$ finishes, $\mathcal{ONL}$ has accepted $x_j$ jobs of weight $2^{j-1}$, for $j = 1, 2, \ldots, i-1$. At least $(x_1 - m)/2^{i-1} + (x_2 - m)/2^{i-2} + \ldots + (x_{i-1} - m)/2$ jobs are in an interval $I_i = [t_i, t_i + D_i + 1)$. In round $i$, $m(D_i + 1)$ new jobs, each of weight $2^{i-1}$, arrives at time $t_i$ with deadline $t_i + D_i + 1$. $\mathcal{OPT}$ can fill all the slots in $I_i$ with the new weight-$2^{i-1}$ jobs, fill all slots in in $I_{i-1} - I_i$ with weight-$2^{i-2}$ jobs, and so on, and fill all slots in $I_1 - I_2$ with weight-1 jobs. $\mathcal{OPT}$ therefore gets a profit of

$$
\begin{aligned}
& m(2 + D_2) + m(2 + D_3)(2) + \ldots + m(2 + D_i)2^{i-2} + m(1 + D_i)2^{i-1} \\
&= m((2 + D_2) + 2(2 + D_3) + \ldots + 2^{i-2}(2 + D_i) + 2^{i-1}(2 + D_i) - 2^{i-1}) \\
&= m((i + 1)(1 + D/2) - 2^{i-1}) \text{ (since } (2 + D_{j+1})2^{j-1} = 1 + D/2 \text{ for any } j) \\
&> m((i + 1)D/2 - 2^{i-1})
\end{aligned}
$$

So if $\mathcal{ONL}$ accepts $x_i$ of these new jobs, it must be that $x_1 + 2x_2 + \ldots + 2^{i-1}x_i \geq (m/R)((i+1)D/2 - 2^{i-1})$. Now, the interval $[t_i + 1, t_i + D_i + 1)$ can be partitioned into two halves, and one of them will contain at least $(x_1 - m)/2^i + \ldots + (x_i - m)/2$ accepted jobs. Let $I_{i+1} = [t_{i+1}, t_{i+1} + D_{i+1} + 1)$, where $t_{i+1} \in \{t_i + 1, t_i + 2 + D_{i+1}\}$, be this denser interval. Proceed to the next round.

If after some round $i$ we have $x_1/2^i + x_2/2^{i-1} + \ldots + x_i/2 - m > mD/2^i$, i.e. $x_1 + 2x_2 + \ldots + 2^{i-1}x_i > m(D + 2^i)$, then as $mD/2^i > mD_{i+1}$ and $1/2^i + 1/2^{i-1} + \ldots + 1/2 < 1$, this would imply $(x_1 - m)/2^i + (x_2 - m)/2^{i-1} + \ldots + (x_i - m)/2 > mD_{i+1}$, which means there would not have been enough timeslots to accept all these jobs for it to stay $R$-competitive, so $\mathcal{ONL}$ loses. We show that this must happen after some finite number of rounds. In fact, we already have the constraints that for all $i$, $x_1 + 2x_2 + \ldots + 2^{i-1}x_i \geq (m/R)((i+1)D/2 - 2^{i-1})$, so all we need to prove is that there exists a finite $i$ such that $(m/R)((i+1)D/2 - 2^{i-1}) > m(D + 2^i)$. This inequality is true if and only if $i > 2R(1 + 2^i/D) + 2^i/D - 1$. Note that we can choose $D \gg 2^i$, so the right hand side in the expression above is at most $2R(1 + \epsilon) + \epsilon - 1$ for some arbitrarily small $\epsilon > 0$. So when $i = 2R$, i.e. after $2R$ rounds, $\mathcal{ONL}$ must lose. By choosing $R = \log \rho / 2$, we have $W = 2^{i-1} = 2^{2R-1} < \rho$, so indeed $W < \rho$ as stated. □

For one machine, the following shows that the $O(\rho)$ upper bound in Theorem 1 is asymptotically optimal.

**Theorem 5.** *No deterministic algorithm has a competitive ratio better than* $\max(\rho + 1, 2)$ *in the immediate decision model with one machine.*

## 3 Immediate Notification

### 3.1 Upper bound

Clearly, the algorithm for immediate decision can also be applied in the immediate notification model. Here we give a similar algorithm that utilizes the ability to reschedule accepted jobs and give a slightly smaller competitive ratio (which is more significant when $m = 1$ and $\rho$ is small). Again we maintain a provisional schedule $S$ (even though in IMM NOTIF the algorithm is not required to announce this information upfront), and define $S'$ similarly.

**Algorithm 3.** Let $\beta = (2\rho + 1)^{1/m}$. We again maintain the property that $S'(i, u) \geq \beta S'(i + 1, u)$. For each newly arriving job $j$ arriving at time $t$, run the following admission procedure:

- Step 1. Find the earliest time $u$, where $u < d(j)$, such that $w(S'(1, u)) \leq w(j)/\beta$. If no such $u$ exists then reject $j$. Otherwise, remove the job $k = S'(m, u)$, i.e. the smallest-weight job committed to time $u$, accept $j$, and assign the slot vacated by $k$ to $j$. Note that now $j$ becomes the new $S'(1, u)$. If $k$ is a null job, the procedure finishes. Otherwise we say that $k$ is *displaced* by $j$, and we proceed to Step 2.

– Step 2. Find the earliest time $u$ such that $u < d(k)$ and $w(k) > w(S'(1,u))$. If no such $u$ exists then $k$ is evicted and the procedure finishes. Otherwise remove the job $k' = S'(1,u)$ from $S$, replace it with $k$, and then this $k'$ becomes the new $k$ and Step 2 is repeated.

Intuitively, after $k$ is removed in Step 1, it tries to reinsert itself back into $S$ by following a similar procedure, except that this time $\beta$ effectively becomes 1 and that we only uses $S'(1,u)$, i.e. the heaviest jobs, not the lightest ones. At the end of this chain of displacements, there may eventually be a job that will not find a place in $S$. We call it the job *evicted by* $j$. Note that when we refer to evicted jobs, it is the last job in a chain of displaced jobs, not the job directly displaced by $j$'s insertion.

We show that this algorithm is also $O(m(1+\rho)^{1/m})$-competitive with $\beta = (2\rho+1)^{1/m}$; however for $m = 1$ and $\rho$ small it gives some improvement over Algorithm 1; see Table 1.

**Theorem 6.** *Algorithm 3 is $O(m(1+\rho)^{1/m})$-competitive with $\beta = (2\rho+1)^{1/m}$. When $m = 1$, the competitive ratio becomes $(2\rho + 2 + 2\sqrt{\rho^2+2\rho})$ with $\beta = 1+\rho+\sqrt{\rho^2+2\rho}$.*

In a way similar to Theorem 3 the algorithm can be modified to retain the $O(\log(1+\rho))$-competitiveness when $m$ is large.

### 3.2 Lower bounds

We first consider the case of a single machine. The 1.618 lower bound in [15, 5, 20] for the no-commitment, no-penalty model carries over here. The following theorem gives a stronger lower bound for $\rho > 0$.

**Theorem 7.** *In the immediate notification model with one machine, no deterministic algorithm can give a competitive ratio better than* $\min((1+\rho+\sqrt{(1+\rho)^2+4})/2, 2)$.

The above theorem cannot give any lower bound larger than 2 even if $\rho$ is large. The next theorem gives a lower bound that increases with $\rho$. It only works for large $\rho$, but it works for any number of machines.

**Theorem 8.** *Any deterministic algorithm has competitive ratio $\Omega(\log\rho/\log\log\rho)$ for the immediate notification model with $m$ machines.*

## 4 Separate Arrival and Start Times

Finally, we consider a situation when the arrival time and the earliest allowed start time of a job is not necessarily the same; for example, an order is placed at a certain time but the delivery should only take place during a certain time window that does not begin immediately. There are other situations where this

may be useful; see e.g. [1, 7] where it is even assumed that all job arrivals are completed before execution takes place, i.e. they are separate processes.

Formally, a job is represented by a release time $r(j)$, its earliest possible start time $s(j)$, its deadline $d(j)$, and its weight $w(j)$, where $r(j) \leq s(j) < d(j)$. All of the algorithms in this paper can be straightforwardly adapted to this model with the same competitive ratio and essentially identical proof, which we omit. At first sight it might appear this model is easier as it gives 'advance notice' of jobs for the online algorithm. However, it turns out that in this model we can prove tighter (in fact, optimal) lower bounds. The separation of arrival and starting times allows job sequences to behave like the 'online-list' model (where jobs arriving in the same time step are presented to the online algorithm one by one, the next only after the previous has been dealt with), in contrast to the 'online-time' model where all jobs arriving at the same time are presented together. (For a discussion of these models see e.g. [19].) In the case with penalties the online-list model is disadvantageous to the online algorithm.

In IMM NOTIF there is a gap between the upper bound $O(\log \rho)$ and the lower bound $\Omega(\log \rho / \log \log \rho)$. If we separate arrival and start times however we can close this gap by giving a matching lower bound:

**Theorem 9.** *For the immediate mnotification model with separate arrival and starting times, any deterministic online algorithm is $\Omega(\log \rho)$-competitive.*

Note that in [1] the case where items for auction are considered. In the case where the constraints are represented by a uniform matroid of rank $m$, it can be interpreted as a special case of our scheduling problem where all jobs have the same $s(j)$ and $d(j) = s(j)+1$, all arrivals are well before $s(j)$ and jobs arrive one by one. The items correspond to the $m$ slots offered by $m$ machines in a single time step. They give matching upper and lower bounds of $1+2\rho+2\sqrt{\rho(1+\rho)} = \Theta(\rho)$ when $m = 1$. Our results here (Theorems 2 and 9) can be seen as results for $m > 1$.

## References


1. Babaioff, M., Hartline, J. D. and Kleinberg, R. D.: Selling ad campaigns: online algorithms with cancellations. In *Proc. 10th ACM Conference on Electronic Commerce*, 61–70, 2009.
2. Bunde, D. P. and Goldwasser, M. H.: Dispatching equal-length jobs to parallel machines to maximize throughput. In *Proc. 12th Scandinavian Workshop on Algorithm Theory*, LNCS 6139, 346–358, 2010.
3. Chen, S., Tong, L. and He, T.: Optimal deadline scheduling with commitment. In *49th Annual Allerton Conference on Communication, Control and Computing*, 111–118, 2011.
4. Chin, F. Y. L., Chrobak, M., Fung, S. P. Y., Jawor, W., Sgall, J. and Tichý, T.: Online competitive algorithms for maximizing weighted throughput of unit jobs. *Journal of Discrete Algorithms*, 4(2):255–276, 2006.
5. Chin, F. Y. L. and Fung, S. P. Y.: Online scheduling with partial job values: Does timesharing or randomization help? *Algorithmica*, 37(3):149–164, 2003.

6. Cole, R., Dobzinski, S. and Fleischer, L.: Prompt mechanisms in online auctions. In *Proc. 1st Symposium on Algorithmic Game Theory*, LNCS 4997, 170–181, 2008.
7. Constantin, F., Feldman, J., Muthukrishnan, S. and Pal, M.: An online mechanism for ad slot reservations with cancellations. In *Proc. 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1265–1274, 2009.
8. Ding, J., Ebenlendr, T., Sgall, J. and Zhang, G.: Online scheduling of equal-length jobs on parallel machines. In *Proc. 15th European Symposium on Algorithms*, LNCS 4698, 427–438, 2007.
9. Ding, J. and Zhang, G.: Online scheduling with hard deadlines on parallel machines. In *Proc. 2nd International Conference on Algorithmic Aspects in Information and Management*, LNCS 4041, 32–42, 2006.
10. Ebenlendr, T. and Sgall, J.: A lower bound for scheduling of unit jobs with immediate decision on parallel machines. In *Proc. 6th Workshop on Approximation and Online Algorithms*, LNCS 5426, 43–52, 2008.
11. Englert, M. and Westermann, M.: Considering suppressed packets improves buffer management in QoS switches. In *Proc. 18th ACM-SIAM Symposium on Discrete Algorithms*, 209–218, 2007.
12. Fung, S. P. Y.: Online preemptive scheduling with immediate decision or notification and penalties. In *Proc. 16th International Computing and Combinatorics Conference*, LNCS 6196, 389–398, 2010.
13. Goldwasser, M. H.: A survey of buffer management policies for packet switches. *SIGACT News*, 45(1):100–128, 2010.
14. Goldwasser, M. H. and Kerbikov, B.: Admission control with immediate notification. *Journal of Scheduling*, 6:269–285, 2003.
15. Hajek, B.: On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time. In *Proc. Conference on Information Sciences and Systems*, 434–438, 2001.
16. Kaminsky, P. and Hochbaum, D.: Due date quotation models and algorithms. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* (Leung, J. Y.-T. and Henderson, J. H. eds.), Chapter 20, Chapman and Hall/CRC, 2004.
17. Keskinocak, P., Ravi, R. and Tayur, S.: Scheduling and reliable lead-time quotation for orders with availability intervals and lead-time sensitive revenues. *Management Science* 47(2), 264–279, 2001.
18. Kesselman, A., Lotker, Z., Mansour, Y., Patt-Shamir, B., Schieber, B. and Sviridenko, M.: Buffer overflow management in QoS switches. In *Proc. 33th ACM Symposium on Theory of Computing*, 520–529, 2001.
19. Pruhs, K., Sgall, J. and Torng, E.: Online Scheduling. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* (Leung, J. Y.-T. and Henderson, J. H. eds.), Chapter 15, Chapman and Hall/CRC, 2004.
20. Zhu, A.: Analysis of queueing policies in QoS switches. *Journal of Algorithms*, 53:137–168, 2004.