Algorithmica (2011) 59: 569–582
DOI 10.1007/s00453-009-9321-4

Competitive Algorithms for Due Date Scheduling

Nikhil Bansal · Ho-Leung Chan · Kirk Pruhs

Received: 23 May 2008 / Accepted: 30 April 2009 / Published online: 9 May 2009

Abstract We consider several online scheduling problems that arise when customers request make-to-order products from a company. At the time of the order, the company must quote a due date to the customer. To satisfy the customer, the company must produce the good by the due date. The company must have an online algorithm with two components: The first component sets the due dates, and the second component schedules the resulting jobs with the goal of meeting the due dates.

The most basic quality of service measure for a job is the *quoted lead time*, which is the difference between the due date and the release time. We first consider the basic problem of minimizing the average quoted lead time. We show that there is a $(1+\epsilon)$-speed $O(\frac{\log k}{\epsilon})$-competitive algorithm for this problem (here k is the ratio of the maximum work of a job to the minimum work of a job), and that this algorithm is essentially optimally competitive. This result extends to the case that each job has a weight and the objective is weighted quoted lead time.

We then introduce the following general setting: there is a non-increasing profit function $p_i(t)$ associated with each job J_i. If the customer for job J_i is quoted a due date of d_i, then the profit obtained from completing this job by its due date is $p_i(d_i)$. We consider the objective of maximizing profits. We show that if the company must

The research of K. Pruhs was supported in part by NSF grants CNS-0325353, CCF-0448196, CCF-0514058 and IIS-0534531.

N. Bansal
IBM T.J. Watson Research, P.O. Box 218, Yorktown Heights, NY, USA
e-mail: nikhil@us.ibm.com

H.-L. Chan
Max-Planck-Institut für Informatik, Saarbrücken, Germany
e-mail: hlchan@mpi-inf.mpg.de

K. Pruhs (✉)
Computer Science Department, University of Pittsburgh, Pittsburgh, PA, USA
e-mail: kirk@cs.pitt.edu

finish each job by its due date, then there is no $O(1)$-speed poly-log-competitive algorithm. However, if the company can miss the due date of a job, at the cost of forgoing the profits from that job, then we show that there is a $(1+\epsilon)$-speed $O(1+1/\epsilon)$-competitive algorithm, and that this algorithm is essentially optimally competitive.

Keywords Online algorithms · Scheduling · Competitive analysis · Due dates · Supply chain

"As a strategic weapon, time is the equivalent of money, productivity, quality, even innovation."
George Stalk, Boston Consulting Group

1 Introduction

We consider several online scheduling problems that arise when customers request make-to-order products from a time-based competitive company. As an example, consider the Atlas Door company, whose product was industrial doors that come in an wide variety of widths, heights, and materials. Traditional companies manufactured doors in batches, and stored the resulting doors in large warehouses. Atlas built just-in-time flexible factories, investing extra money to buy tooling to reduce change-over times. Traditionally when customers order a door from the manufacturer, the customers usually had to wait a week for a response as to when their order could be filled. (This wait would be even longer if the door was not in stock or scheduled for production.) Atlas invested in an efficient front-end that automated the order entry, pricing and scheduling process. Atlas could price and schedule almost all of its orders immediately. Atlas was able to charge a premium for rush orders since Atlas knew these orders could not be met by its competitors. As a result, in ten short years Atlas went from start-up company to supplying 80% of the distributors in the US [10]. Similar success stories for other time-based competitors, such National Bicycle and Lutron electronics, can be found in [3].

In this paper, we formulate some basic online due-date scheduling problems. All of the problems that we consider share the following basic framework. Jobs arrive to the system online over time. Job J_i arrives at its release time r_i. Job J_i has a work requirement w_i, which is the time it would take to complete the job on a unit speed machine. At the time r_i, the system must quote a due date d_i for job J_i to the customer submitting the job. We assume that the scheduler may preempt jobs (and later restart them from the point of preemption). It is easy to see that preemption is necessary to avoid worst case scenarios where many short jobs arrive just after a long job has been started. In our due-date setting, an online algorithm has to have two components: a component that sets the due date, and a component that schedules the jobs. Intuitively, setting the due dates presents a dilemma to the online algorithm. Earlier due dates, if not missed, will make the customer happier, and presumably increase profit for the company. On the other hand, setting due dates to be too early restricts the scheduler from setting earlier due dates on later arriving jobs. The main goal of this paper is to gain some understanding about when and how an online algorithm can reasonably cope with this dilemma.

We use traditional worst-case competitive analysis, and more generally consider the resource augmentation setting. Recall that an online scheduling algorithm A is s-speed c-competitive for an objective function f if for all input instances $f(A_s(I)) \leq c \cdot f(OPT_1(I))$, where $A_s(I)$ is the output of algorithm A with an s speed processor on input I, and $OPT_1(I)$ is the optimal unit speed schedule for input I [4]. Instances that generally arise in practice for many scheduling problems have the following threshold property: The performance of the system, with an optimal scheduling algorithm, is reasonable when the load is bounded away from some threshold, but degrades precipitously as the load exceeds the threshold. Intuitively, a $(1+\epsilon)$-speed c-competitive algorithm should perform reasonably well on such common instances since it would then have essentially the same threshold as the optimal algorithm. For more information on resource augmentation and its motivation, see the surveys [8, 9].

Quoted Lead Time The standard quality of service measure for a job J_i is the *quoted lead time (QLT)*, which is defined to be $d_i - r_i$ [5, 6]. Probably the most natural related objective is to minimize the average, or equivalently total, quoted lead times of the jobs under the constraint that all jobs must be finished by their due date. That is, the objective is $\sum(d_i - r_i)$. It is instructive to consider the case that the online scheduler can delay the setting of due dates. In particular, consider the most extreme case where the online scheduler can set the due date to be the completion time of a job. The quoted lead time problem then becomes the standard flow time problem, for which the online Shortest Remaining Processing Time algorithm is optimal. Similarly, the weighted quoted lead time problem becomes the classic problem of minimizing the weighted flow time. For our problem, the requirement that the due dates be set immediately makes the resulting online problems only more difficult, since the offline scheduler can still set the due date equal to the completion time of a job. In particular, it turns out that even in the unweighted case, any (1-speed) algorithm is at least $\Omega(\sqrt{k})$ competitive where k is the ratio of the maximum to minimum job size.

We thus consider the problem in the resource augmentation setting. Our first main result is a $(1+\epsilon)$-speed $O(\frac{\log k}{\epsilon})$-competitive algorithm, which we call BIT, for the quoted lead time problem. More generally, this result extends to the weighted case where each job has a weight and the objective is weighted quoted lead time. The parameter k is the ratio of the maximum density of any job to the minimum density of any job, where the density of a job is its weight divided by its size. The BIT algorithm is a composition of three well known scheduling algorithms: Highest Density First (or equivalently Shortest Job First in the unweighted setting), Round Robin, and First-Come-First-Served. We also show that this guarantee is essentially the best possible for any algorithm. In particular, we show that for any $c > 1$, any c-speed algorithm must be at least $\Omega(\log k/c)$ competitive. This lower bound holds even in the unweighted case.

Arbitrary Profit Functions Minimizing total quoted lead times is a reasonable objective function if a company wishes to promise generally fast response to a collection of essentially equivalent customers/jobs. But in some situations, say if the company charges a premium for rush orders, the company may explicitly know how much

profit it can obtain from a particular due date. For example, companies such as Atlas Door charge additional shipping and handling for rush orders. In these cases, this known profit should be incorporated into the objective function. We introduce the following general setting: there is a non-increasing profit function $p_i(t)$ associated with each job J_i. If the customer for job J_i is quoted a due date of d_i, then the profit obtained from completing this job by its due date is $p_i(d_i)$. We consider the objective of maximizing profits, that is the objective is $\sum p_i(d_i)$. Note that the online scheduler in this problem has the power to reject jobs since setting the due date to $+\infty$ is essentially equivalent to rejecting the job.

Consider the following dilemma for the online scheduler: after scheduling a low-value long job, several emergency jobs arrive that will yield high profit if early due dates are set. However, setting early due dates for these emergency jobs would make it infeasible to finish both the emergency jobs and the low-value job by their due dates. The company would like to be able to drop the low-value job to make greater profit from the emergency jobs. We get two different models depending on whether this dropping is allowed. In the *reliable* model, the scheduler must complete every job by its due date. In this model, the company could not get profit from these emergency jobs in our example instance. In the *unreliable* model, the company merely does not earn any profit from those jobs that it does not complete by their due dates. Many companies offer no recourse to a customer that is not provided a service by/at the time promised other than that they are not charged. If you have had a flight canceled by Ryan Air, or a package not delivered in time by Federal Express, you probably have had first hand experience with this.

Our results on profit maximization are given in Sect. 3. We show that in the reliable model there is no $O(1)$-speed poly-log-competitive algorithm. In the unreliable model, we show that there is a $(1+\epsilon)$-speed $O(1)$-competitive algorithm, and we show that this algorithm is essentially optimally competitive. These results match the intuition that the ability to break promises is crucial to a company that wants to maximize profit. Once again it is instructive to consider relaxing the problem so that the online algorithm can delay the setting of the due dates until the completion time of job. In this relaxation, there is no difference between the reliable and unreliable models. A special case of this relaxed problem that has previously been studied in the literature is if the profit functions are constant until some time/deadline, and then zero thereafter. In this case, the goal becomes to maximize the profit of jobs that are completed by their deadlines. For this case, it is known that resource augmentation is necessary [1] and the best result that is achievable is $O(1+\epsilon)$-speed $O(1)$-competitiveness [4]. Thus $O(1+\epsilon)$-speed $O(1)$-competitiveness is the best possible result that we could have hoped for in our due date profit maximization problem. So for profit maximization, the introduction of due dates makes the resulting online problem significantly harder only in the reliable case.

Related Previous Work Within the world of supply chain management there is an extensive literature on due date scheduling problems. Surveys of due date scheduling problems, with hundreds of references, can be found [5] and [6]. As always with scheduling problems, there is a vast number of reasonable formulations that have been studied. The majority of the literature uses experimentation as the method of

algorithm evaluation. There is also a fair amount of literature that uses queuing-theory analysis to evaluate online algorithms.

There is to our knowledge only one previous paper, namely [7], in the literature on online due-date scheduling problems that uses worst-case analysis. In this special case, all jobs have the same work, say w. If the due date of a job is set to be as early as feasible, namely at $r_i + w$, then the profit obtained from this job is a fixed constant ℓ. For every unit of time that the due date is delayed, one unit of profit is lost. It is easy to see that the online algorithm, that accepts a job if it can gain positive profit, has a competitive ratio of $\Omega(\ell)$. This algorithm may accept many jobs with low profit, and thus not be able to reap profit from later arriving jobs. Reference [7] shows that the reliable online algorithm that rejects jobs on which it won't earn at least 61.8% of the maximum profit ℓ, is 1.618-competitive. Note that this result relies heavily on both the fact that the profit function has a very special structure, and on the fact that all jobs are identical except release times. The paper [7] considers other special cases and variations, for example, if there are two different kinds of jobs instead of one and if quotations can be delayed.

2 Minimizing Weighted Quoted Lead Time

This section considers the problem of minimizing weighted quoted lead time. Recall that each job J_i has release time r_i, amount of work w_i and weight c_i. An online algorithm needs to set a due date d_i when J_i is released and the quoted lead time (or simply lead time) of J_i is $\ell_i = (d_i - r_i)$. The online algorithm must complete each J_i by its deadline d_i and objective is to minimize $\sum c_i \ell_i$, i.e., the total weighted lead time. We define the *density* of a job J_i to be c_i/w_i. Let k be the ratio of the maximum to minimum density of the jobs. We give a simple algorithm BIT that is $(1+\epsilon)$-speed $O((\log k)/\epsilon)$-competitive and we show that BIT already achieves a nearly optimal competitive ratio.

2.1 The Algorithm BIT and Its Analysis

Let us first give some motivation for BIT. For any job sequence I, let L be the minimum possible total weighted lead time and let F be the minimum possible total weighted flow time. We note that $L = F$, because the total weighted lead time is at least the total weighted flow time and they can be equal when the due date of each job is set to its completion time.

Consider the algorithm Highest Density First (HDF) that at any time works on the highest density job. It is known that HDF is a good strategy for the objective of weighted flow time.

Lemma 1 [2] *For any $\epsilon > 0$, HDF is $(1+\frac{\epsilon}{2})$-speed $(1+\frac{2}{\epsilon})$-competitive for minimizing weighted flow time.*

Suppose, BIT runs a copy of HDF; furthermore, whenever a job J_i is released, the due date d_i is set to at most α times the completion time of J_i in HDF, assuming

that no more jobs will be released. If it turned out that all jobs were completed by their deadlines, it would imply that BIT is $\alpha(1+\frac{2}{\epsilon})$-competitive for total weighted lead time. Of course, the problem is that HDF may not complete J_i by d_i since many higher density jobs might arrive during $[r_i, d_i]$. Interestingly, it turns out that by giving BIT slightly extra speed, and by choosing α large enough, we can guarantee that each job will be completed by its due date. We define BIT formally as follows.

We may assume without loss of generality that the minimum density of a job is 1. Increasing the weight of jobs by a factor of at most 2, we assume that all jobs have densities 2^j for $j = 1, 2, \ldots, \log k$. BIT divides the jobs into classes $C_1, C_2, \ldots, C_{\log k}$ where all jobs in C_j have density 2^j. BIT operates as follows.

Setting Due Dates When a job J_i of class C_j is released at time r_i, let $w(C_j)$ be the amount of remaining work for jobs in class C_j at time r_i immediately before the release of J_i. BIT sets the due date d_i to $r_i + (w(C_j) + w_i) \cdot \frac{2\log k}{\epsilon}$. Note that if no more jobs are released, a processor of speed $\frac{\epsilon}{2\log k}$ can complete J_i and all the jobs in C_j by d_i.

Processing Jobs BIT divides its processing power into two parts by time-sharing. Thus, we may assume that BIT is using a processor P_1 of speed $(1+\frac{\epsilon}{2})$ and also another processor P_2 of speed $\frac{\epsilon}{2}$.

- P_1 runs HDF, i.e., P_1 always processes the class with the highest density. With this class, it works on the job with the earliest release time.
- P_2 is evenly time-shared among all the $\log k$ classes. For each class C_j, it processes the job in C_j with the earliest release time using speed $\frac{\epsilon}{2\log k}$.

Observation 1 BIT *completes each job J_i by its due date d_i.*

Proof Suppose J_i belongs to C_j. Since jobs in C_j are dedicated a processor of speed $\frac{\epsilon}{2\log k}$ and we run the jobs within a class in the order of their arrival times, the job J_i will be completed by d_i irrespective of the jobs that arrive after r_i. □

We are now ready to bound the lead time ℓ_i of each job J_i in BIT. For any $s \geq 1$, Let HDF(s) denote a stand-alone copy of HDF using a s-speed processor. Let f_i be the flow time of a job J_i in the schedule of HDF$(1+\frac{\epsilon}{2})$.

Lemma 2 *For any job J_i, $\ell_i \leq \frac{2\log k}{\epsilon} \cdot (1+\frac{\epsilon}{2}) \cdot f_i$.*

Proof Let J_i be a job of class C_j and let $w_h(C_j)$ be the amount of unfinished work under HDF$(1+\frac{\epsilon}{2})$ for jobs in class C_j at time r_i. Since BIT is also running a copy of HDF$(1+\frac{\epsilon}{2})$, it must be that $w(C_j)$, the unfinished work for class C_j jobs under BIT is at most $w_h(C_j)$. Hence f_i is at least $(w_h(C_j)+w_i)/(1+\frac{\epsilon}{2})$. Therefore,

$$\ell_i = (w(C_j)+w_i)\cdot\frac{2\log k}{\epsilon} \leq (w_h(C_j)+w_i)\cdot\frac{2\log k}{\epsilon} \leq \frac{2\log k}{\epsilon}\cdot\left(1+\frac{\epsilon}{2}\right)\cdot f_i. \quad \square$$

Theorem 3 *For any $\epsilon > 0$,* BIT *is $(1+\epsilon)$-speed $\frac{4\log k}{\epsilon}(1+\frac{\epsilon}{2})(1+\frac{2}{\epsilon})$-competitive. The competitive ratio can improved to $\frac{16\log k}{\epsilon}$ for $\epsilon \geq 2$.*

Proof Let *Opt* be the adversary that minimizes the total weighted lead time for the modified job sequence. Then,

$$\begin{aligned}\sum c_i \ell_i &\leq \frac{2\log k}{\epsilon}\left(1+\frac{\epsilon}{2}\right)\sum c_i f_i \quad \text{(by Lemma 2)}\\ &\leq \frac{2\log k}{\epsilon}\left(1+\frac{\epsilon}{2}\right)\left(1+\frac{2}{\epsilon}\right)\\ &\quad \times \text{total weighted flow time of } \mathit{Opt} \quad \text{(by Lemma 1)}\\ &= \frac{2\log k}{\epsilon}\left(1+\frac{\epsilon}{2}\right)\left(1+\frac{2}{\epsilon}\right) \times \text{total weighted lead time of } \mathit{Opt}.\end{aligned}$$

Since the weights of the jobs are increased by BIT by at most a factor of two to ensure that job densities are powers of two, the total weighted lead time of *Opt* is at most two times that of the optimal one for the original job sequence. This completes the first part of the proof.

For $\epsilon \geq 2$, we note that HDF$(1+\frac{\epsilon}{2})$ is $\frac{4}{1+\epsilon/2}$-competitive on weighted flow time (because HDF$(1+\frac{\epsilon}{2})$ has weighted flow time at most $\frac{2}{1+\epsilon/2}$ times that of HDF(2), and HDF(2) is 2-competitive on weighted flow time). As above, we obtain that for $\epsilon \geq 2$, the total weighted lead time of BIT is at most $\frac{2\log k}{\epsilon}(1+\frac{\epsilon}{2})(\frac{4}{1+\epsilon/2}) = \frac{8\log k}{\epsilon}$ times that of *Opt*. Since the weights increased at most by a factor of two to ensure that job densities are powers of 2, the result follows. □

2.2 Lower Bounds

We show that BIT already achieves a nearly optimal competitive ratio and also a lower bound when no speed-up is given.

Theorem 4 *Let $c > 1$ be any integer. Any deterministic c-speed algorithm is $\Omega(\frac{\log k}{c})$-competitive. This holds even for unweighted instances.*

Proof Consider any c-speed algorithm A. Let $x \geq 2$ be any integer. We will release a sequence of at most $(c\log x + 1)$ batches of jobs. All jobs have weight 1. Batch B_i has 2^{i-1} jobs and each job has size $1/2^{i-1}$. First, batch B_1 is released at time 0. For $i = 2, \ldots, c\log x + 1$, if at least half of the jobs in B_{i-1} have due date at most $\frac{\log x}{2}$, then batch B_i is released immediately at time 0; otherwise, the job sequence terminates. Note that if batch $B_{c\log x+1}$ is released, there must be at least $\frac{c\log x}{2}$ units of work with due date at most $\frac{\log x}{2}$. A has a c-speed processor and needs to meet the due date of the jobs, so A needs to set the due date of all jobs in batch $B_{c\log x+1}$ to be greater than $\frac{\log x}{2}$.

Let B_r be the last batch of jobs released. Note that A sets the due date of at least $\frac{2^{r-1}}{2}$ jobs to be at least $\frac{\log x}{2}$. The total weighted lead time of A is $\Omega(2^{r-1}\log x)$.

The adversary can schedule the jobs in reverse order of arrival, giving an objective of at most $\sum_{i=1}^{r} 2^{i-1}(r+1-i) = O(2^{r-1})$. Thus, A is $\Omega(\log x)$-competitive. Note that the densities of the jobs range from 1 to $2^{r-1} \leq 2^{c \log x}$. The theorem follows as $k \leq 2^{c \log x}$ and $\log x \geq \frac{\log k}{c}$. □

Theorem 5 *Any deterministic algorithm using a unit-speed processor is $\Omega(\sqrt{k})$-competitive. This holds even for unweighted instances.*

Proof Consider any algorithm A. We release a big job J_0 at time 0 with $c_0 = 1$ and $w_0 = k$. If the due date set by A is at least $k\sqrt{k}$, then A is already $\Omega(\sqrt{k})$-competitive. Otherwise, for $i = 1, 2, \ldots, k\sqrt{k}$, a small job J_i is released with $r_i = i - 1$, $c_i = 1$ and $w_i = 1$. Since A needs to process J_0 before $k\sqrt{k}$, A must set the due dates of at least k small jobs to be greater than $k\sqrt{k}$, and thus for these jobs $\sum c_i(d_i - r_i) \geq \frac{k^2}{2}$. On the other hand, the adversary can set the due date of J_0 to be $k\sqrt{k}+k$, and the due date of each small job J_i to be i. It gives a value of $k\sqrt{k} + k\sqrt{k} + k$ for the objective function. Thus, A is $\Omega(\sqrt{k})$-competitive. □

3 Profit Maximization

We assume in this section that each job J_i has an associated non-increasing profit function $p_i(t)$ specifying the profit obtained if the due date is set to time t, and our objective is the total profit obtained from the jobs finished by their due date. Our main result, which we give in Sect. 3.1, is a $(1+2\delta)$-speed $(6 + \frac{12}{\delta} + \frac{8}{\delta^2})$-competitive algorithm in the unreliable model where the scheduler is not obligated to finish all jobs by their due date. As mentioned earlier, resource augmentation is necessary to obtain reasonable bounds as this problem is as at least as hard as maximizing the profit of jobs completed by their deadlines [1]. In Sect. 3.2 we also show that every deterministic c-speed algorithm is at least $(1 + \frac{1}{c \cdot 2^c})$-competitive, even when the job size and profit are bounded. In Sect. 3.3, we consider the reliable version of the problem where all jobs must be completed by their due date. We show that every deterministic c-speed algorithm is $\Omega(k^{1/c})$-competitive.

3.1 The Algorithm for the Unreliable Model

This section describes our algorithm INT and shows that it is $(1+2\delta)$-speed $(6 + \frac{12}{\delta} + \frac{8}{\delta^2})$-competitive in the unreliable model, for any $\delta > 0$. INT maintains a pool P of jobs and only processes jobs in P. Whenever a job J_i is released, INT assigns a due date d_i to J_i (using the procedure below), and J_i is put into P if $p_i(d_i) > 0$. A job J_i remains in P until it is completed or becomes non-viable (i.e., the remaining work is more than $(1+2\delta)$ times the duration between the current time and its due date). Note that once the due date of J_i is fixed, we can unambiguously define $p_i = p_i(d_i)$ as the profit and $u_i = p_i/w_i$ as the density of J_i.

Intuitively, INT runs the highest density job available. And intuitively INT sets the due date to be the earliest time t so that if no more jobs arrive, and always the highest density job is run, then even a slightly slower processor could finish the job

by t. Unfortunately, it requires some complications to make the idea precise and to get the technical details to work out.

Let $c > 1 + \frac{1}{\delta}$ be a constant (that we set later to $1 + \frac{2}{\delta}$). Throughout, we assume that INT runs at speed $1 + 2\delta$ and the optimum offline algorithm runs at speed 1. We now formally specify how the due dates are set, and which job is executed at each time t.

Setting Due Dates INT maintains an invariant that each job admitted in P is associated with a collection of time intervals $I(J_i) = \{[t_1, t'_1], [t_2, t'_2], \ldots, [t_h, t'_h]\}$, where $r_i \le t_1 < t'_1 < t_2 < \cdots < t_h < t'_h = d_i$. This collection $I(J_i)$ is specified (and fixed) as soon as J_i is released. The total length of the intervals in $I(J_i)$ is $\frac{1+\delta}{1+2\delta} w_i$. Roughly speaking, J_i is expected to be run during $I(J_i)$. Note that the total length is $(1 + \delta)$ times more than the time required to run J_i.

When a new job J_i is released at r_i, INT tests whether a time $t' > r_i$ is a good choice for due date as follows: Assuming t' is the due date, then the profit of J_i is $p_i(t')$ and density $u_i = p_i(t')/w_i$. Let $X(\frac{u_i}{c})$ be the set of jobs in P with density at least $\frac{u_i}{c}$. Consider the time interval $[r_i, t']$ and the associated intervals of jobs in $X(\frac{u_i}{c})$. Let $A = \{[a_1, a'_1], [a_2, a'_2], \ldots, [a_h, a'_h]\}$ be the maximal subintervals of $[r_i, t']$ that do not overlap with the associated intervals of any job in $X(\frac{u_i}{c})$. We say that t' is a *feasible due date* if the total length of intervals in A is at least $\frac{1+\delta}{1+2\delta} w_i$. Note that a feasible due date always exists by choosing t' large enough. INT sets the due date of J_i to be the earliest time t' such that t' is a feasible due date. If $p_i(t') > 0$, then J_i is put into P and $I(J_i)$ is set to the corresponding A. We wish to point that a job could have arbitrarily many associated intervals.[1]

Executing Jobs At any time t, let S be the set of jobs J_i in P that are allowed to run at t (i.e. all J_i such that $I(J_i)$ contains some interval $[t_j, t'_j]$ and $t \in [t_j, t'_j]$). INT processes the job in S with highest density.

We first state two observations about INT before presenting the analysis.

Observation 2 *At any time t, let J_1 and J_2 be two jobs in P such that some intervals of $I(J_1)$ and $I(J_2)$ are overlapping. Then, either $u_1 > c \cdot u_2$ or $u_2 > c \cdot u_1$.*

Proof Follows directly from the procedure for defining $I(J_1)$ and $I(J_2)$. □

Consider the overall execution of INT in hindsight for the sequence of jobs. For any time t, let S be the set of jobs J_i ever put into P such that $I(J_i)$ contained some interval $[t_j, t'_j]$ and $t \in [t_j, t'_j]$. The density of t is defined as the density of the highest density job in S.

[1] As an example, consider a sequence of n unit-size job, each with density 1.1 or 0.9, released alternatively at time 0. The associated intervals of jobs with density 1.1 are $[2i, 2i + 1]$, while that for jobs with density 0.9 are $[2i + 1, 2i + 2]$, for $i = 0, 1, \ldots$. Assume then a job J_i of size $n/2$ and density $u_i = c$ is released. INT will set the associated intervals of J_i as $A = \{[2i + 1, 2i + 2]$ for $i = 0, 1, \ldots\}$.

Observation 3 *Consider any job J_i and a time $t \geq r_i + w_i$. Suppose* INT *sets the due date of J_i to be strictly greater than t. Let $u_i = p_i(t)/w_i$ and let L be the amount of time during $[r_i, t]$ such that the density is at least $\frac{u_i}{c}$. Then, $L \geq \frac{\delta}{1+2\delta}(t - r_i)$.*

Proof If L is less than $\frac{\delta}{1+2\delta}(t - r_i)$, then t is a feasible due date for J_i and INT would have set the due date of J_i to be at most t, which yields a contradiction. □

We now turn our attention to analyzing this algorithm. Let C be the set of jobs completed by INT, and R be the set of jobs that have ever been put into P. For any set X of jobs, let $\|X\|$ be the total profit of jobs in X according to the due dates set by INT. We first lower bound the profit obtained by INT.

Lemma 6 *For C and R as defined above, $\|C\| \geq (1 - \frac{1}{\delta(c-1)})\|R\|$, or equivalently, $\|R\| \leq \frac{\delta(c-1)}{\delta(c-1)-1}\|C\|$.*

Proof We use a charging scheme to prove the lemma. For each job J_i in C, we give p_i units of credit to J_i initially. The remaining jobs in $R - C$ are given 0 units of credit initially. We will describe a method to transfer the credits such that at the end, each job $J_i \in R$ has credit at least $(1 - \frac{1}{\delta(c-1)})p_i$, which completes the proof.

The method to transfer credit is as follows. At any time t, let S be the set of jobs that have an associated interval containing t. Let J_i be the highest density job in S. Then, for each other job J_j in S, J_i transfers credit to J_j at a rate of $(\frac{1+2\delta}{\delta})u_j$.

We first show that every job J_j in R receives credit at least p_j either initially or transferred from other jobs. This clearly holds for jobs in C. For any job J_j in $R - C$, as J_j could not be completed during $I(J_j)$, it must have received credit for at least $\frac{\delta}{1+2\delta} \cdot w_j$ units of time. Thus, the total credit obtained is at least $(\frac{\delta}{1+2\delta})w_j \cdot (\frac{1+2\delta}{\delta})u_j = w_j u_j = p_j$.

We now show that the credit transferred out of each job J_i is at most $\frac{1}{\delta(c-1)}p_i$. When a job J_i is the highest density job in S, by Observation 2 the remaining jobs in S have geometrically decreasing densities and hence their total density is at most $\frac{1}{c-1}u_i$. Therefore, the rate of credit transferring out of J_i is at most $(\frac{u_i}{c-1})(\frac{1+2\delta}{\delta})$. Since J_i is the highest density job for at most $\frac{w_i}{1+2\delta}$ units of time, the total credit transferred out is at most of J_i is at most $(\frac{u_i}{c-1})(\frac{1+2\delta}{\delta}) \cdot (\frac{w_i}{1+2\delta}) = \frac{1}{\delta(c-1)}p_i$. □

Next, we upper bound the profit obtained by the adversary. Let A be the set of jobs completed by the adversary. For any set of jobs X, let $\|X\|^*$ be the total profit of jobs in X according to the due dates set by the adversary. We may assume that the adversary only completes jobs with non-zero profit. Let A_1 be the set of jobs in A such that the due date set by INT is no later than that by the adversary. Let $A_2 = A \setminus A_1$. Then, the total profit obtained by the adversary is $\|A\|^* = \|A_1\|^* + \|A_2\|^* \leq \|A_1\| + \|A_2\|^* \leq \|R\| + \|A_2\|^*$. Note that $\|A_1\| \leq \|R\|$ because each job in A_1 must lie in R since INT set a due date for it with non-zero profit. We now bound the profit of jobs in A_2.

For any $u > 0$, let $T(u)$ be the total length of time that the adversary is running jobs in A_2 with density at least u (where the density is determined according to the

due dates set by the adversary). For the schedule of INT, let $L(\frac{u}{c})$ be the total length of time such that the density is at least $\frac{u}{c}$. (Recall that the density of a time t is the density of the highest density job that has an associated interval containing t.)

Lemma 7 *For every $u > 0$, $T(u) \leq \frac{2(1+2\delta)}{\delta} L(\frac{u}{c})$.*

Proof For any job $J_i \in A_2$, let the span of J_i be the time interval $[r_i, d_i^*]$, where d_i^* is the due date set by the adversary. For any $u > 0$, let $A_2(u)$ be the set of jobs in A_2 with density at least u. Consider the union of spans of all jobs in $A_2(u)$. It may consist of a number of disjoint time intervals, and let ℓ be its total length. Clearly, $T(u) \leq \ell$.

Let $M \subseteq A_2$ be the minimum cardinality subset of A_2 such that the union of spans of jobs in M equals that of A_2. Note that the minimality property implies no three jobs in M have their spans overlapping at a common time. This implies that we can further partition M into M_1 and M_2 such that within M_1 (resp. M_2), any two jobs have disjoint spans. Now, either M_1 or M_2 has total span of length at least half of that of M. Without loss of generality, suppose that it is M_1. Note that each interval in M_1 corresponds to a span of some job in A_2. Applying Observation 3 to each such interval, it follows that the density of INT is at least $\frac{u}{c}$ for at least $\frac{\delta}{1+2\delta}$ fraction of time during the intervals of M_1. Thus, $L(\frac{u}{c}) \geq \frac{\delta}{2(1+2\delta)} \cdot T(u)$, which completes the proof. □

Let $\{\phi_1, \phi_2, \ldots, \phi_m\}$ be the set of densities of jobs in A_2 (determined by the adversary's due dates), where $\phi_i > \phi_{i+1}$ for $i = 1, \ldots, m-1$. For simplicity, let $\phi_0 = \infty$ and $\phi_{m+1} = 0$. For $i = 1, \ldots, m$, let ℓ_i be the length of time that the adversary is running jobs of density ϕ_i. Similarly, for $i = 1, \ldots, m$, let α_i be the length of time that INT has density in the range $[\phi_i/c, \phi_{i-1}/c)$. Then, the following holds.

Lemma 8 *Let K be a constant. If $T(u) \leq K \cdot L(\frac{u}{c})$ for every $u \geq 0$, then $\sum_{i=1}^{m} \ell_i \phi_i \leq K \cdot \sum_{i=1}^{m} \alpha_i \phi_i$.*

Proof Rephrasing $T(\phi_i) \leq K \cdot L(\frac{\phi_i}{c})$ in terms of ℓ_i and α_i, we obtain the following inequalities for each $i = 1, \ldots, m$

$$\ell_1 + \cdots + \ell_i \leq K(\alpha_1 + \cdots + \alpha_i).$$

Multiplying the ith inequality by $(\phi_i - \phi_{i+1})$ (which is strictly positive for all i) and adding them, we obtain the desired result that $\sum_{i=1}^{m} \ell_i \phi_i \leq K \cdot (\sum_{i=1}^{m} \alpha_i \phi_i)$. □

Lemma 9 $\|A\|^* \leq (1 + \frac{2(1+\delta)c}{\delta}) \|R\|$.

Proof Recall that $\|A\|^* = \|A_1\|^* + \|A_2\|^* \leq \|R\| + \|A_2\|^*$. By Lemmas 7 and 8,

$$\|A_2\|^* \leq \sum_{i=1}^{m} \ell_i \phi_i \leq \frac{2(1+2\delta)}{\delta} \sum_{i=1}^{m} \alpha_i \phi_i. \tag{1}$$

Let q_i be the total profit for jobs whose density in INT is in the range of $[\phi_i/c, \phi_{i-1}/c)$. For any job J_j, as the total length of the associated intervals is $\frac{1+\delta}{1+2\delta} w_j$, it follows that $\alpha_i \frac{\phi_i}{c} \leq \frac{1+\delta}{1+2\delta} \cdot q_i$. Combining with (1), we obtain that

$$\|A_2\|^* \leq \frac{2(1+2\delta)}{\delta} \sum_{i=1}^{m} \alpha_i \phi_i \leq \frac{2(1+\delta)c}{\delta} \sum_{i=1}^{m} q_i \leq \frac{2(1+\delta)c}{\delta} \|R\|$$

which implies the desired result. □

Theorem 10 *For any $\delta > 0$, the algorithm described above is $(1+2\delta)$-speed, $O(1)$-competitive for profit maximization in the unreliable model. In particular $\|A\|^* \leq (6 + \frac{12}{\delta} + \frac{8}{\delta^2})\|C\|$.*

Proof The result follows by Lemmas 6 and 9, and setting $c = 1 + \frac{2}{\delta}$. □

3.2 Lower Bounds in the Unreliable Model

It is easily seen that resource augmentation is necessary to obtain $O(1)$ competitive algorithms in the unreliable model. In fact, by the results of Baruah et al. [1] it follows that every deterministic algorithm is $\Omega(k)$ competitive even if all the profit functions are of the type $p_i(t) = p_i$ during $[0, d_i]$ and 0 thereafter. Here k is the ratio of the maximum to minimum job density. We now show that 1-competitiveness is not possible for any online algorithm, even when it is given faster processors and the job size and profit are bounded.

Theorem 11 *Let $c \geq 1$ be any integer. Consider the profit maximization problem in the unreliable model. Any deterministic c-speed algorithm is at least $(1 + \frac{1}{c \cdot 2^c})$-competitive, even when all jobs are of size 1 and the profit is either 0 or in the range* [0.5, 1].

Proof For any c-speed algorithm A, we construct a sequence of at most $c+1$ jobs defined as follows. Let $\alpha = 1/2^c$. A job J_1 is released with $r_1 = 0$, $w_1 = 1$, and $p_1(t) = 1$ for $t \in [0, 1]$ and $p_1(t) = 1 - \alpha$ otherwise. For $i = 2, 3, \ldots, c+1$, J_i is released if A sets d_{i-1} to be at most 1; in that case, J_i is released with $r_i = 0$, $w_i = 1$, and $p_i(t) = 1$ for $t \in [0, 1]$ and $p_i(t) = 1 - 2^{i-1}\alpha$ otherwise. Note that if J_{c+1} is released, it means that A sets d_i to be at most 1 for $i = 1, \ldots, c$. If A sets d_{c+1} to be at most 1, then A must miss the due date of some job; else (i.e., A sets d_{c+1} to be greater than 1), then the profit for J_{c+1} is $1 - 2^c \cdot \alpha = 0$.

Let J_r be the last job released. If A sets d_r to be greater than 1, then the total profit of A is at most $(r-1) + (1 - 2^{r-1} \cdot \alpha) = r - 2^{r-1}\alpha$. The adversary can set the due date of J_r to be 1 and set the due date of $J_1, \ldots, J_{r-1}$ to be greater than 1. Note that by setting the due dates far enough, the adversary can complete each job $J_1, \ldots, J_r$. The total profit of the adversary is $\sum_{i=1}^{r-1}(1 - 2^{i-1}\alpha) + 1 = r - (2^{r-1} - 1)\alpha$. Thus, the competitive ratio of A is at least $\frac{r - 2^{r-1}\alpha + \alpha}{r - 2^{r-1}\alpha} \geq 1 + \frac{1}{c \cdot 2^c}$. Else if A sets d_r to be at most 1, then $r = c + 1$ in this case. A using a c-speed processor cannot complete

all the $c+1$ jobs, so its total profit is at most c. The adversary can set the due date of J_{c+1} to be 1 and due date of $J_1, \ldots, J_c$ to be greater than 1. The total profit is $\sum_{i=1}^{c}(1-2^{i-1}\alpha)+1 = c+1-(2^c-1)\alpha = c+\alpha$. Thus, the competitive ratio of A is at least $\frac{c+\alpha}{c} = 1+\frac{1}{c\cdot 2^c}$. □

3.3 The Reliable Model

We show substantially stronger lower bounds for the profit maximization problem in the reliable model where jobs must be completed by their due dates. Let Δ be the ratio of the maximum to minimum job size. Let $p_i^* = p_i(w_i)$ be the maximum possible profit achievable by a job J_i, and set $u_i^* = p_i^*/w_i$. Let k denote the maximum to minimum ratio of u_i^*. The following lower bound states that $O(1)$-competitive algorithm is possible only when both k and Δ are constant.

Theorem 12 *Any deterministic online algorithm is at least $\Omega(k\cdot\Delta)$-competitive in the reliable model of the profit maximization problem.*

Proof Consider any algorithm A. A job J_1 is released with $r_1 = 0$ and $w_1 = 1$. The profit function $p_1(t)$ equals 1 for $t \in [0, 1]$, and equals 0 otherwise. A must set d_1 to 1; otherwise, the competitive ratio is unbounded. Then, J_2 is released with $r_2 = 0$ and $w_1 = \Delta$. The profit function $p_2(t)$ equals $k\cdot\Delta$ for $t \in [0, \Delta]$, and equals 0 otherwise. A must set d_2 to be at least $\Delta+1$ as it needs to complete J_1. Thus, A is at least $(k\cdot\Delta)$-competitive. □

Next we show that constant competitive ratio is not possible even if we use an arbitrarily large constant speed-up.

Theorem 13 *Let $c \geq 1$ be an integer. Any deterministic c-speed algorithm is $\Omega(k^{1/c})$-competitive in the reliable model of the profit maximization problem.*

Proof For any c-speed algorithm A, we release a sequence of at most $c+1$ jobs defined as follows. Let $x \geq 2$ be an integer. A job J_1 is released with $r_1 = 0$, $w_1 = 1$, and $p_1(t) = x$ for $t \in [0, 1]$ and $p_1(t) = 0$ otherwise. For $i = 2, 3, \ldots, c+1$, J_i is released if A sets d_{i-1} to be at most 1; in that case, J_i is a job with $r_i = 0$, $w_i = 1$, and $p_i(t) = x^i$ for $t \in [0, 1]$ and $p_i(t) = 0$ otherwise. Note that if J_{c+1} is released, it means that A sets d_i be at most 1 for $i = 1, \ldots, c$. To meet these due dates, A must set d_{c+1} greater than 1.

Let J_r be the last job released. Note that A sets d_r to be greater than 1. The total profit of A is at most $(x^{r-1}+x^{r-1}+\cdots+x) \leq \frac{x^r}{x-1}$. The adversary can set the due date of J_r to be 1 and obtain a profit of x^r. Thus, A is at least $(x-1)$-competitive. Note that the density ratio k is at most x^c, that is, $x \geq k^{1/c}$. □

4 Conclusions

As best as we can tell, this paper is the first competitive analysis of reasonably general due date scheduling problems. For the total lead time minimization problem, we

found that the introduction of due dates made the task of the online scheduler significantly harder. For the profit maximization problem, we found that the task of the online scheduler became significantly more difficult in the reliable model, but not in the unreliable model. It would be interesting to investigate other due date scheduling problems, from say the surveys [5] and [6], using worst-case analysis to get a better understanding of the effect of the introduction of due dates.

Acknowledgement We would like to thank Steef van de Velde for helpful discussions.

References

1. Baruah, S., Koren, G., Mishra, B., Raghunathan, A., Rosier, L., Shasha, D.: On-line scheduling in the presence of overload. In: Symposium on Foundations of Computer Science, pp. 100–110. IEEE Comput. Soc., Los Alamitos (1991)
2. Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Pruhs, K.: Online weighted flow time and deadline scheduling. J. Discrete Algorithms **4**(3), 339–352 (2006)
3. Fisher, M.: What is the right supply chain for your product. Harvard Bus. Rev. **75**, 105–116 (1997)
4. Kalyanasundaram, B., Pruhs, K.: Speed is as powerful as clairvoyance. J. ACM **47**(4), 617–643 (2000)
5. Kaminsky, P., Hochbaum, D.: Due date quotation models and algorithms. In: Leung, J.Y.-T. (ed.) Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, Boca Raton (2004). Chap. 20
6. Keskinocak, P., Tayur, S.: Due date management policies. In: Simchi-Levi, D., Wu, S.D., Shen, Z.-J. (eds.) Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era, pp. 485–554. Springer, Berlin (2004)
7. Keskinocak, P., Ravi, R., Tayur, S.: Scheduling and reliable lead-time quotation for orders with availability intervals and lead-time sensitive revenues. Manag. Sci. **47**(2), 264–279 (2001)
8. Pruhs, K.: Competitive online scheduling for server systems. SIGMETRICS Perform. Eval. Rev. **34**(4), 52–58 (2007)
9. Pruhs, K., Sgall, J., Torng, E.: Online scheduling. In: Leung, J.Y.-T. (ed.) Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, Boca Raton (2004)
10. Stalk, G.: Time—the next source of competitive advantage. Harvard Bus. Rev. **66**, 41–51 (1988)