Ann Oper Res manuscript No.
(will be inserted by the editor)

Online Production Planning to Maximize the Number of On-Time Orders

Nicholas G. Hall · Marc E. Posner · Chris N. Potts

Received: date / Accepted: date

Abstract We consider a production planning problem with two planning periods. Detailed planning occurs in the first period, when complete information is known about a set of orders that are initially available. An additional set of orders becomes available at the start of the second planning period. The objective is to maximize the number of on-time orders. We derive an upper bound on the competitive ratio of any deterministic online algorithm, relative to the performance of an algorithm with perfect information about the second set of orders. This ratio depends on the relative lengths of the two planning periods. We also describe an efficient algorithm that delivers a solution which asymptotically achieves this upper bound ratio as the number of jobs becomes large.

Keywords Online Planning · Scheduling · Planning Horizon · Competitive Analysis

1 Introduction

We consider a production planning environment with two planning periods, an initial short-term period and a future long-term period. Complete information is known about the orders that are initially available for processing. Consequently, detailed production planning is performed during the first period. However, the possibility of new orders arriving at the end of the first period complicates this

Nicholas G. Hall
Fisher College of Business, The Ohio State University, Ohio 43210, United States
E-mail: hall.33@osu.edu

Marc E. Posner
Department of Integrated Systems Engineering, The Ohio State University, Ohio 43210, United States
E-mail: posner.1@osu.edu

Chris N. Potts
School of Mathematical Science, University of Southampton, Southampton SO17 1BJ, UK
E-mail: C.N.Potts@soton.ac.uk

planning. Information about new orders becomes available when they arrive at the start of the long-term planning period.

A producer only accepts and schedules an order that can be completed by its due date. Thus, the order is rejected if it does not complete on time. Hence, a natural objective is to find a production plan that maximizes the number of accepted orders. This is also a standard measure of customer service in make-to-order manufacturing environments (Pinedo, 2012).

Our work originates from both the planning horizon and online scheduling literatures. Applications of the related planning horizon literature can be found in inventory management (Ignall and Veinott, 1969), production planning (McClain and Thomas, 1977), and scheduling and sequencing (Bean et al., 1991). The planning environment that we consider is representative of periodic ordering and production processes that are widely used in industry (Reid and Sanders, 2005; Russell and Taylor, 2006). Within such processes, until an order arrives, no details are known about its properties, specifically its processing time and due date. Chand et al. (2002) consider multiperiod production planning problems where data for future periods has a diminishing effect on current decisions. In this situation, a decision maker can work effectively with a limited set of data from the present and the near future.

A more specific application of our model occurs when a company uses pre-orders for their newly developed products. A pre-order is defined as "an order for an item that has not yet been released" (Zhao and Stecke, 2010). The use of pre-orders is desirable for customers, because they are assured of receiving a product, in many cases with personalized customizations. It also benefits the seller by allowing the demand for a new product to be estimated from preorder sales, so that an appropriate price can be found (Li and Zhang, 2013). In many cases, companies limit the number of pre-orders that they accept, so that reasonable customer service levels can be provided at and after the release date. The uncertainty over the amount and variety of regular orders creates difficulty in deciding how many pre-orders to accept. Thus, focusing on the worst case, as is usual in the competitive analysis of online algorithms (Pruhs et al., 2004), is an appropriate approach to this problem for risk-averse manufacturers.

Online planning period scheduling problems, where orders can only arrive at known future times, are introduced by Hall et al. (2009). For the problem of minimizing the weighted sum of completion times, they describe a best possible online algorithm. If all orders are available at the start of the planning horizon, then the algorithm of Moore (1968) finds an optimal schedule for the number of on-time orders problem. However, if the orders have different release dates, then this problem is unary *NP*-hard (Lenstra et al., 1977). The preemptive version of the problem with release dates is solved optimally by an efficient algorithm (Lawler, 1983; Baptiste, 1999). A fully online version, where preempted orders must be restarted, is studied by Hoogeveen et al. (2000). Zheng et al. (2007) and Fung (2008) consider generalizations where preemption of an order incurs a penalty cost that is proportional to the order's value. Special cases of the fully online problem with equal processing times are discussed by Chrobak et al. (2006) and Zheng et al. (2008).

We develop an upper bound on the performance of any algorithm for the online planning period problem. We then describe an efficient online algorithm, and show that it has asymptotically the best possible bound of any deterministic online algorithm.

This paper is organized as follows. Section 2 provides our notation and a formal description of the problem studied. Section 3 derives an upper bound on the performance of any online algorithm. In Section 4, we describe an efficient online scheduling algorithm for the problem. In Section 5, we show that our algorithm is asymptotically best possible. Finally, Section 6 contains a conclusion and some suggestions for future research.

2 Preliminaries

In this section, we provide our notation and a formal description of the problem. We also develop some preliminary results that are used in subsequent sections.

2.1 Notation and problem description

We consider an online planning period scheduling problem in which job orders, hereafter *jobs*, are processed nonpreemptively on a single machine. There are two planning periods defined by the intervals $[0, t]$ and $[t, T]$, where t and T are known at time zero, and $0 < t < T$.

For problem instance I, a set of jobs $N_0(I)$ is released at time zero. Each job $j \in N_0(I)$ has a due date d_j, where $0 < d_j \leq T$. A decision is made about which jobs from $N_0(I)$ are scheduled to start before time t. Once started, a job must be processed to completion, i.e., preemption is not allowed. At time t, a new set of jobs $N_t(I)$ is released. Then, a second decision is made about which jobs are to be processed by time T. Each job $j \in N_t(I)$ has a due date $d_j = T$. Let $N(I) = N_0(I) \cup N_t(I)$ denote the set of all jobs. Also, each job $j \in N(I)$ has a positive processing time (or size) p_j. When clear from the context, we omit "(I)" from the description.

Let $S \subseteq N$ denote the set of on-time jobs in some schedule. The objective is to maximize the number of jobs that are processed by their due dates. Any job that is not processed by its due date is rejected and does not appear in the schedule.

For instance I, let $z^{\mathcal{A}}(I)$ denote the objective function value obtained by using the schedule generated by some deterministic online Algorithm $\mathcal{A}$. Also, let $z^*(I)$ denote the number of on-time jobs in an optimal offline solution. We refer to $z^{\mathcal{A}}(I)/z^*(I)$ as the *performance ratio* of Algorithm $\mathcal{A}$ for instance I. The *competitive ratio* of Algorithm $\mathcal{A}$, $R^{\mathcal{A}}(n)$, is the infimum of $z^{\mathcal{A}}/z^*$ over all problem instances with at least $n = |N_0|$ jobs, where N_0 is a reduced set of jobs that results from the preprocessing procedure given in Section 2.2. Also, let $R^*(n)$ denote the competitive ratio of a best possible online algorithm with at least $n = |N_0|$ jobs.

In this paper, some specific job sequences are used. In an *SPT sequence* (Smith 1956), the jobs appear in nondecreasing processing time order. In an *LPT sequence* (Graham 1966), the jobs appear in nonincreasing processing time order. Also, in an *EDD sequence* (Jackson 1955), the jobs appear in nondecreasing due date order. For consistency, we assume that ties are broken by the smaller job index.

2.2 Preliminary results

Given an initial set N_0 of jobs, the algorithm of Moore (1968) is used to discard a subset of jobs from N_0. Moore's algorithm successively appends jobs in EDD sequence to the current partial schedule, which is initially empty. Whenever there is a due date violation, a job in the partial schedule with the largest processing time is discarded from the partial schedule.

Theorem 1 *Given any problem instance, there exists an optimal offline schedule where no job from $N_0(I)$ that is discarded by Moore's algorithm is on time.*

Proof Moore's algorithm finds a schedule for the jobs of N_0 with a maximum number of on-time jobs. Let N_t^* denote the jobs of N_t that are on time in an optimal offline schedule σ^*. Since each job of N_t has due date T, without loss of generality we assume that the jobs of N_t^* are processed after all other jobs in σ^*. As a result, if we modify the due date of each job $j \in N_0$ to be $\min\{d_j,\ T - \sum_{j \in N_t^*} p_j\}$, then Moore's algorithm finds an optimal schedule. Further, Moore's algorithm does not select a previously discarded job. $\square$

Theorem 1 shows that the jobs discarded by Moore's algorithm do not affect the value of z^*. Now, suppose that instance I has $N_0 = \{1\}$, where $p_1 = T$. If some Algorithm $\mathcal{A}$ discards job 1, then suppose that $N_t = \emptyset$ in instance I. This yields $z^{\mathcal{A}}/z^* = 0/1 = 0$. Alternatively, if Algorithm $\mathcal{A}$ processes job 1, then let N_t contain identical jobs of size $(T-t)/|N_t|$. Hence, $z^{\mathcal{A}}/z^* = 1/|N_t| \to 0$ as $|N_t| \to \infty$, and $R^{\mathcal{A}}(1) = 0$ for this instance, where $|N_0| = 1$. Extending this instance by constructing $n-1$ duplicates of job 1 shows that if no job of N_0 is processed, then $z^{\mathcal{A}}/z^* = 0/1 = 0$. Alternatively, if one if these jobs is processed, then $z^{\mathcal{A}}/z^* = 1/|N_t| \to 0$ as $|N_t| \to \infty$. Thus, without any preprocessing of the set N_0, we would obtain $R^{\mathcal{A}}(n) = 0$, for $n \geq 1$, which does not allow meaningful competitive analysis to be undertaken. Consequently, we assume that the jobs of N_0 are preprocessed by Moore's algorithm, and n is the number of jobs that remain in N_0 after preprocessing. Further, it is reasonable to use n as a baseline for the number of on-time jobs, since processing more than n jobs of N_0 so that they complete by their due dates is impossible.

For the remainder of this paper, we restrict our attention to problem instances for which all the jobs of N_0 are on time if scheduled in EDD order. For convenience, we assume that the jobs of $N_0 \cup N_t$ are indexed in EDD order, where $i \in N_0$ and $j \in N_t$ implies that $i < j$. Also, for $i, j \in N_0$ or $i, j \in N_t$, if $i < j$ and $d_i = d_j$, then $p_i \leq p_j$.

We use the following notation associated with a schedule $\sigma^\beta(I)$ for some arbitrary instance I, where β is empty for a schedule $\sigma(I)$ found by Moore's algorithm, $\beta = \mathcal{A}$ for a schedule $\sigma^{\mathcal{A}}(I)$ found by some Algorithm $\mathcal{A}$, and $\beta = *$ for an optimal offline schedule $\sigma^*(I)$:

$E^\beta(I)$ = set of jobs of $N_0(I)$ that start before time t in $\sigma^\beta(I)$
$L^\beta(I)$ = set of jobs of $N_0(I)$ that start no earlier than time t in $\sigma^\beta(I)$
$S^\beta(I)$ = set of jobs that complete on time in σ^β.

When there is no ambiguity, we drop "(I)"; for example $E^{\mathcal{A}}(I)$ becomes $E^{\mathcal{A}}$. Also, we define the number of jobs of $E(I)$ that are discarded in schedule $\sigma^\beta(I)$ by

$$g^\beta(I) \ = \ |E(I) \setminus E^\beta(I)|.$$

In our analysis, we first find an upper bound on the value of R^*, the competitive ratio of a best possible online algorithm for an arbitrarily large value of $n = |N_0|$. Then, we construct an online algorithm whose competitive ratio approaches this value from below as $n \to \infty$. These two results jointly establish that

$$R^* = \begin{cases} \frac{(1+\alpha)^2}{(1+\alpha)^2+\alpha^2}, & \text{if } T/t < 2 \\ \frac{(1+\gamma)^2}{(1+\gamma)^2+1}, & \text{if } T/t \geq 2 \end{cases}, \tag{1}$$

where

$$\alpha = \left(\frac{T-t}{T}\right)^{1/2}, \quad \text{and} \quad \gamma = \left(\frac{T}{t}\right)^{1/2}. \tag{2}$$

Remark 2 $R^* \geq (1+\sqrt{2})^2)/[(1+\sqrt{2})^2+1] = (2+\sqrt{2})/4$.

Proof The first expression for R^* in (1) is decreasing in α, while the second expressing is increasing in γ. Thus, a minimum value of R^* is achieved when $T/t = 2$. Substituting $\sqrt{2}$ for γ in (1) gives the desired result. □

At time t, complete information is known. Applying Moore's algorithm at time t provides an optimal completion of any selection of $E^{\mathcal{A}}$ for Algorithm $\mathcal{A}$. Consequently, the behavior of $\mathcal{A}$ is characterized by $g^{\mathcal{A}}(I)$, the number of jobs of E that are discarded. If too many jobs are discarded from E, then a worst-case scenario occurs when $N_t = \emptyset$. Alternatively, if not enough jobs from E are discarded, then worst-case behavior occurs when a set of new jobs arrive at time t. By removing more of the longer jobs of E than $\sigma^{\mathcal{A}}$, and replacing them with shorter jobs of L, more jobs of N_t can potentially be processed by σ^* in the interval $[t, T]$. Thus, an effective online algorithm should achieve a suitable balance when deciding on the jobs of E to discard.

Let

$$g_0(n) \equiv g_0 = n(1 - R^*). \tag{3}$$

Since $R^* = (n - g_0)/n$, the target number of jobs that an online algorithm should aim to discard from E is given by $\min\{g_0, |E|\}$.

3 Upper Bound Analysis

In this section, we provide an upper bound on the competitive ratio of any online scheduling algorithm. We first describe an upper bound that depends only on n.

Theorem 3 *$R^{\mathcal{A}}(n) \leq (n-1)/n$ for any online Algorithm $\mathcal{A}$.*

Proof For some small $\epsilon > 0$, consider an instance where $p_j = (t-\epsilon)/(n-1)$ and $d_j = t$ for $j = 1, \ldots, n-1$, $p_n = T - t + \epsilon$ and $d_n = T$. Thus, $E = \{1, \ldots, n\}$ and $L = \emptyset$. If Algorithm $\mathcal{A}$ processes jobs $1, \ldots, n$, then no time remains for scheduling additional jobs. Consider the completion of the partial schedule where N_t contains a large number of short jobs. The result is a performance ratio that is arbitrarily close to 0. Alternatively, if Algorithm $\mathcal{A}$ discards at least one job, then suppose $N_t = \emptyset$. Because the optimal schedule processes all jobs, $R^{\mathcal{A}}(n) \leq (n-1)/n$. □

When T/t is close to one or large, the competitive ratio $(n-1)/n$ in Theorem 3 is tight. Intuitively, when T/t is close to one, there is relatively little time in the second planning period in which the online algorithm can make poor choices. Alternatively, when T/t is large, complete information becomes available relatively close to time zero. We next show that, for intermediate values of T/t, the competitive ratio is smaller than $(n-1)/n$.

An online algorithm benefits by discarding long jobs of E and replacing them with short jobs of L. This fills the interval $[0, t]$ with many short jobs and provides flexibility to process the jobs of N_t. An optimal offline algorithm has two advantages. First, it does not have to protect against $N_t = \emptyset$. When this occurs, it does not discard any jobs. Second, when $N_t \neq \emptyset$, the offline algorithm can discard as many jobs as needed to accommodate the shorter jobs of N_t. Consequently, it does not benefit as much as an online procedure when the job sizes within sets E, L, and N_t vary. Hence, as we establish, there exists a family of dominating instances where all jobs within each of E, L, and N_t have the same size, and the performance ratio approaches R^* as n tends to infinity.

To establish an upper bound on the competitive ratio, we evaluate a particular set of instances that have the following characteristics.

Property 1: n_E long jobs, each released at time zero with size e and due date t, where $n_E e = t$.

Property 2: n_L short jobs, each released at time zero with size ℓ and due date T, where $e > \ell$ and $n_L \ell = \min\{T, 2t\} - t$.

Property 3: $n_t \in \{0, \lfloor (T-t)/\ell \rfloor\}$ short jobs, each released at time t with size ℓ and due date T.

We refer to instances of this type as (e, ℓ)-instances. To describe a particular (e, ℓ)-instance, it is necessary to specify a value of n where $n = n_E + n_L$, and also how many jobs of n are assigned to n_E and n_L. Then, Properties 1 and 2 are used to find the processing times e and l of the long and short jobs, respectively.

For an (e, ℓ)-instance, an optimal offline schedule, σ^*, processes without interruption $\min\{n_E, \lfloor [T - \ell(n_L + n_t)]/e \rfloor\}$ long jobs starting at time 0. At the completion of these jobs, all n_L short jobs followed by all n_t short jobs are processed.

We compare σ^* to a schedule $\sigma^{\mathcal{A}}$ generated by some online algorithm $\mathcal{A}$ where s_E early (long) and s_L late (short) jobs are processed in the interval $[0, t]$. Further, there is insufficient time to complete any additional jobs in this interval. The remaining $n_L - s_L$ short jobs and as many as possible of the n_t short jobs that are released at time t are processed in the interval $[t, T]$.

If no additional jobs are released at time t, then the performance ratio is

$$\frac{s_E + n_L}{n_E + n_L}. \tag{4}$$

Alternatively, if $n_t = \lfloor (T-t)/\ell \rfloor$, then the performance ratio for this scenario is

$$\frac{s_E + s_L + n_t}{\max\{\lfloor [T - \ell(n_L + n_t)]/e \rfloor, 0\} + n_L + n_t}. \tag{5}$$

To provide an upper bound on the competitive ratio of any online algorithm, we consider the two cases $T/t < 2$ and $T/t \geq 2$ separately.

Case A: $T/t < 2$.

Based on the limiting case as $n \to \infty$, consider an (e, ℓ)-instance where

$$n_L = \lceil n\alpha \rceil, \tag{6}$$

and where α is defined in (2). The other parameters that we deduce from the three defining properties of an (e, ℓ)-instance are $n_E = n - n_L$, $e = t/n_E$ and $\ell = (T - t)/n_L$. A straightforward computation shows that the required condition $e > \ell$ is satisfied.

Given any choice of s_E by the online scheduling algorithm, a corresponding value of s_L is determined by evaluating the maximum number of shorter jobs that can fit in the remainder of the interval $[0,\ t]$. This yields

$$s_L = \lfloor (t - s_E e)/\ell \rfloor. \tag{7}$$

Next, we define

$$K_1 = \frac{(1+\alpha)^2}{(1+\alpha)^2 + \alpha^2}, \tag{8}$$

where K_1 is the value of R^* in (1) when $T/t < 2$.

To derive an upper bound on the competitive ratio of any online algorithm for Case A, we need the following result.

Lemma 4 *If $n_L = \lceil n\alpha \rceil$, then*

$$\frac{n_L + (n^2\alpha^2 - nn_L)(K_1 - \alpha)/(n - n_L) + \alpha^2}{n - 2n\alpha^2 + n_L} \le \frac{n\alpha - n\alpha(K_1 - \alpha) + \alpha^2}{n - 2n\alpha^2 + n\alpha}.$$

Proof

$$\begin{aligned}
&\frac{n\alpha - n\alpha(K_1 - \alpha) + \alpha^2}{n - 2n\alpha^2 + n\alpha} - \frac{n_L + (n^2\alpha^2 - nn_L)(K_1 - \alpha)/(n - n_L) + \alpha^2}{n - 2n\alpha^2 + n_L} \\
&= \frac{(n_L - n\alpha)[-(n - 2n\alpha^2) + (K_1 - \alpha)n(n + n\alpha - n\alpha^2 - 2n\alpha^3 + n_L\alpha)/(n - n_L) + \alpha^2]}{(n - 2n\alpha^2 + n\alpha)(n - 2n\alpha^2 + n_L)} \\
&= \frac{(n_L - n\alpha)\Big[-(n - \alpha^2) + (1+\alpha)^2/[(1+\alpha)^2 + \alpha^2] - \alpha(n + n\alpha - n\alpha^2 - 2n\alpha^3 + n_L\alpha)/(n - n_L) + \alpha^2\Big]}{(n - 2n\alpha^2 + n\alpha)(n - 2n\alpha^2 + n_L)} \\
&= \frac{(n_L - n\alpha)n[(n_L - n\alpha)(1 + 3\alpha + \alpha^2 - 5\alpha^3 - 6\alpha^4) + n(\alpha + 2\alpha^2 - \alpha^3 - 4\alpha^4 - 2\alpha^5 + 4\alpha^6)]}{(n - 2n\alpha^2\alpha)(n - 2n\alpha^2 + n_L)(n - n_L)[(1+\alpha)^2 + \alpha^2]} \\
&\qquad + \frac{\alpha^2}{(n - 2n\alpha^2 + n\alpha)(n - 2n\alpha^2 + n_L)} \\
&\ge 0.
\end{aligned}$$

The second equality follows from (8). The inequality follows from $n_L = \lceil n\alpha \rceil$ and $\alpha \le \sqrt{2}/2$, and from $1 + 3x + x^2 - 5x^3 - 6x^4 > 0$ and $x + 2x^2 - x^3 - 4x^4 - 2x^5 + 4x^6 > 0$ when $x \in (0,\ \sqrt{2}/2]$. □

We now prove the main result for Case A.

Lemma 5 *If $T/t < 2$, then*

$$R^{\mathcal{A}}(n) < \frac{(1+\alpha)^2}{(1+\alpha)^2+\alpha^2} + \frac{1}{n}$$

for any online Algorithm $\mathcal{A}$.

Proof Consider the (e,ℓ)-instance with parameters defined by equation (6) and the three properties of an (e,ℓ)-instance. Suppose that Algorithm $\mathcal{A}$ chooses $s_E \leq \lfloor n(K_1-\alpha)\rfloor$. If $n_t = 0$, then (4) shows that the performance ratio is

$$\frac{s_E+n_L}{n_E+n_L} < \frac{n(K_1-\alpha)+1+n\alpha}{n} = K_1 + 1/n.$$

The inequality follows from (6) and the assumed upper bound on s_E. Because the competitive ratio is no larger than the performance ratio, the lemma holds for this choice of s_E.

Alternatively, suppose that Algorithm $\mathcal{A}$ chooses $s_E \geq \lfloor n(K_1-\alpha)\rfloor + 1$. If $n_t = n_L$ jobs of size ℓ are released at time t, then (5) establishes that the performance ratio is

$$\begin{aligned}
&\frac{s_E+s_L+n_t}{\max\{\lfloor [T-(n_L+n_t)\ell]/e\rfloor, 0\}+n_L+n_t}\\
&= \frac{s_E+\lfloor (t-s_E e)/\ell\rfloor + n_L}{\max\{\lfloor (T-2n_L\ell)/e\rfloor,\ 0\}+2n_L}\\
&= \frac{\lfloor t/\ell - s_E(e-\ell)/\ell\rfloor + n_L}{\lfloor (2t-T)/e+2n_L\rfloor}\\
&\leq \frac{\lfloor t/\ell - s_E(e-\ell)/\ell\rfloor + n_L + 1}{(2t-T)/e+2n_L}\\
&\leq \frac{t/\ell - n(K_1-\alpha)(e-\ell)/\ell + n_L + 1}{(2t-T)/e+2n_L}\\
&= \frac{tn_L/(T-t) - nn_L(K_1-\alpha)t/((n-n_L)(T-t)) + n(K_1-\alpha)+n_L+1}{(2t-T)(n-n_L)/t+2n_L}\\
&= \frac{n_L(1-\alpha^2)/\alpha^2 - nn_L(K_1-\alpha)(1-\alpha^2)/((n-n_L)\alpha^2) + n(K_1-\alpha)+n_L+1}{(n-n_L)(1-2\alpha^2)/(1-\alpha^2)+2n_L}\\
&= \frac{[n_L+(n^2\alpha^2-nn_L)(K_1-\alpha)/(n-n_L)+\alpha^2](1-\alpha^2)}{(n-2n\alpha^2+n_L)\alpha^2}\\
&\leq \frac{[n\alpha - n\alpha(K_1-\alpha)+\alpha^2](1-\alpha^2)}{(n-2n\alpha^2+n\alpha)\alpha^2}\\
&= \frac{(1-K_1+\alpha)(1+\alpha)}{(1+2\alpha)\alpha} + \frac{1+\alpha}{n(1+2\alpha)}\\
&< K_1 + 1/n.
\end{aligned}$$

The first equality follows from (7) and the choice $n_t = n_L$. The second equality follows from $\ell = (T-t)/n_L$. The first inequality follows from $x/\lfloor y\rfloor < (x+1)/y$

for $0 < x \le \lfloor y \rfloor$. The second inequality follows from $e > \ell$ and from the assumed lower bound on s_E. The third equality follows from $\ell = (T-t)/n_L$ and $e = t/n_E$. The fourth equality follows from the definition of α. The third inequality follows from Lemma 4. □

Case B: $T/t \ge 2$.

An asymptotic analysis similar to that for Case A gives an (e,ℓ)-instance with

$$n_L = \lceil n(1+\gamma)/(2+\gamma) \rceil, \tag{9}$$

where γ is defined in (2). The other parameters, as deduced from the three defining properties of an (e,ℓ)-instance, are $n_E = n - n_L$, $e = t/n_E$ and $\ell = t/n_L$. Similar to Case A, it is straightforward to establish that $e > \ell$.

As in Case A, for any choice of s_E by the online scheduling algorithm, a corresponding value of s_L is given by equation (7).

We now establish the main result for Case B.

Lemma 6 *If $T/t \ge 2$ and $n \ge 5$, then*

$$R^{\mathcal{A}}(n) < \frac{(1+\gamma)^2}{(1+\gamma)^2+1} + \frac{1}{n}$$

for any online Algorithm $\mathcal{A}$.

Proof Consider the (e,ℓ)-instance with parameters defined by equation (9) and the three properties of an (e,ℓ)-instance. Let

$$K_2 = \frac{(1+\gamma)^2}{(1+\gamma)^2+1}.$$

Suppose that Algorithm $\mathcal{A}$ chooses $s_E \le \lfloor n(K_2 - (1+\gamma)/(2+\gamma)) \rfloor$. If $n_t = 0$, then (4) shows that the performance ratio is

$$\frac{s_E + n_L}{n_E + n_L} < \frac{n(1+\gamma)/(2+\gamma) + 1 + n[K_2 - (1+\gamma)/(2+\gamma)]}{n} = K_2 + 1/n.$$

The inequality follows from (9), the relationship $n = n_E + n_L$ and the assumed upper bound on s_E. Hence, the lemma holds for this choice of s_E.

Alternatively, suppose that Algorithm $\mathcal{A}$ chooses $s_E > \lfloor n(K_2 - (1+\gamma)/(2+\gamma)) \rfloor$. If $n_t = \lfloor (T-t)/\ell \rfloor$ jobs of size ℓ are released at time t, then from (5), the performance ratio is

$$\begin{aligned}
&\frac{s_E + s_L + n_t}{\max\{\lfloor [T - (n_L + n_t)\ell]/e \rfloor, 0\} + n_L + n_t} \\
&\quad\le \frac{s_E(\ell - e)/\ell + t/\ell + n_t}{\max\{\lfloor (T - (n_L + n_t)\ell)/e \rfloor,\ 0\} + n_L + n_t} \\
&\quad< \frac{s_E(\ell - e)/\ell + n(1+\gamma)\gamma^2/(2+\gamma)}{[n(1+\gamma)\gamma^2/(2+\gamma)] - 1} \\
&\quad\le \frac{n[K_2 - (1+\gamma)/(2+\gamma)][1 - (\lceil n(1+\gamma)/(2+\gamma) \rceil)/(n - \lceil n(1+\gamma)/(2+\gamma) \rceil)]}{[n(1+\gamma)\gamma^2/(2+\gamma)] - 1}
\end{aligned}$$

$$
\begin{aligned}
& \qquad + \frac{n(1+\gamma)\gamma^2/(2+\gamma)}{[n(1+\gamma)\gamma^2/(2+\gamma)]-1} \\
\leq\ & \frac{n[K_2-(1+\gamma)/(2+\gamma)][1-(n(1+\gamma)/(2+\gamma))/(n-n(1+\gamma)/(2+\gamma))]}{[n(1+\gamma)\gamma^2/(2+\gamma)]-1} \\
& \qquad + \frac{n(1+\gamma)\gamma^2/(2+\gamma)}{[n(1+\gamma)\gamma^2/(2+\gamma)]-1} \\
=\ & \frac{-n[K_2-(1+\gamma)/(2+\gamma)]\gamma + n(1+\gamma)\gamma^2/(2+\gamma)}{[n(1+\gamma)\gamma^2/(2+\gamma)]-1} \\
=\ & \frac{nK_2(\gamma^3+\gamma^2)/(2+\gamma)}{[n(1+\gamma)\gamma^2/(2+\gamma)]-1} \\
<\ & \frac{nK_2(\gamma^3+\gamma^2)/(2+\gamma)+1}{n(1+\gamma)\gamma^2/(2+\gamma)} \\
<\ & K_2 + 1/n.
\end{aligned}
$$

The first inequality follows from (7). The second inequality follows from (9), $n_L = t/\ell$, $n_t = \lfloor (T-t)/\ell \rfloor$, and $e > \ell$. The third inequality follows from (9), the assumed lower bound on s_E, $e = t/(n-n_L)$, and $\ell = t/n_L$. The fourth inequality follows from $K_2 > (1+\gamma)/(2+\gamma)$. The fifth inequality follows from $n \geq 5$ and $K_2 < 1$. The final inequality follows from the fact that $\gamma \geq \sqrt{2}$ implies $(2+\gamma)/(\gamma^2+\gamma^3) < 1$. □

We now provide the main result of this section.

Theorem 7 *When $n \geq 5$, no online algorithm has a competitive ratio that is as large as $R^* + 1/n$.*

Proof The result follows from Lemmas 5 and 6. □

4 An Online Algorithm

In this section, we propose an online scheduling algorithm called Algorithm Balance, for the problem of maximizing the number of on-time jobs processed. We adopt the strategy that no job should be in process at time t because this could have a detrimental effect on scheduling the jobs that arrive at time N_t.

The input for Algorithm Balance includes the value R^*, which is defined in (1). This value is used to compute $g > 0$, which represents the target number of jobs of E to be discarded. The g longest jobs of E are removed from the schedule that is created by Moore's algorithm during preprocessing. These jobs are combined with those in L to form a new set, while the remaining $|E| - g$ jobs of E are scheduled within the time interval $[0, t]$. Jobs of this new set are selected in SPT order to fill as much idle time as possible in the interval $[0, t]$. Then, at time t when full information is available, the schedule is completed by filling the interval $[t, T]$ with as many of the available jobs as possible from $L \cup N_t$.

We now present a formal statement of Algorithm Balance.

Algorithm Balance

0. Input t, T, n, $N_0 = E \cup L$, p_j and d_j for $j \in N_0$, and R^*. Compute $g = \lceil n(1 - R^*) \rceil$.
1. Partition set E into subsets E_1 and E_2, where $|E_2| = g$, $E_1 = E \setminus E_2$ and $p_i \leq p_j$ for all $i \in E_1$ and $j \in E_2$. Set $L' = L \cup E_2$.
2. Partition set L' into subsets L'_1 and L'_2, where $p_i \leq p_j$ for all $i \in L'_1$ and $j \in L'_2$, and $t - p_j < \sum_{i \in E_1 \cup L'_1} p_i \leq t$ for all $j \in L'_2$.
3. Schedule the jobs of $E_1 \cup L'_1$ in EDD order within the interval $[0, t]$.
4. Input N_t, and p_j for $j \in N_t$.
5. Select a maximal set of unscheduled jobs from $(L \cap L'_2) \cup N_t$ in SPT order that fit within the interval $[t, T]$, and schedule these jobs in EDD order starting at time t.

Step 0 of Algorithm Balance selects a value of g such that, if g jobs are discarded and $N_t = \emptyset$, then a performance ratio of R^* is asymptotically achieved. Steps 1–3 schedule jobs within the interval $[0, t]$ by selecting the $|E| - g$ shortest jobs of E to be on-time. Then, the interval $[0, t]$ is filled as much as possible with the shortest remaining jobs of $E \cup L$. Steps 4 and 5 schedule jobs within the interval $[t, T]$ by filling that interval as much as possible with the shortest jobs from those remaining in L and those in N_t.

The running time of Algorithm Balance is $O(n \log n + |N_t|)$ if implemented using a linear time median finding algorithm (Blum et al., 1972) to select jobs of $(L \cap L'_2) \cup N_t$ in Step 5. We use a superscript "$\mathcal{B}$" to refer to the output of this algorithm. For example, $E^{\mathcal{B}}$ is the set of jobs started before time t by Algorithm Balance. Let $g^{\mathcal{B}}(I(n)) = |E(I(n)) \setminus E^{\mathcal{B}}(I(n))|$, where $I(n)$ is an instance with n jobs. Note that $g^{\mathcal{B}}(I(n)) \leq g$, where g is defined in Step 0 of Algorithm Balance.

The competitive analysis of Algorithm Balance that follows in the next section considers a continuous version of the original problem, which we denote as P_C. Specifically, any job can be fractionally processed in P_C and an associated proportional contribution for that job is included in the objective function.

When we consider a schedule for problem P_C, sets E and/or L may contain a fraction of a job. To describe the cardinality of these sets, we let $|\cdot| \in \mathcal{R}_+$ count a job piece based on the proportion that is selected. For example, if $E = \{1, 3, 4_1\}$, where 4_1 is a piece of job 4, then $|E| = 2 + p_{4_1}/p_4$.

We now describe an $O(n \log n + |N_t|)$ time algorithm that provides an optimal solution for Problem P_C.

Algorithm OfflineC

0. Input t, T, $N_0 = E \cup L$, p_j and d_j for $j \in N_0$, and p_j for $j \in N_t$.
1. Schedule the jobs of E in EDD order, followed by the jobs of L in EDD order, followed by the jobs of N_t in arbitrary order. Set $N'_0 = N_0$ and $N'_t = N_t$.
2. Remove jobs j of $N'_0 \cup N'_t$ in LPT order by setting $N'_0 = N'_0 \setminus \{j\}$ or $N'_t = N'_t \setminus \{j\}$ until either $\sum_{j \in N'_0 \cup N'_t} p_j = \min\{T, \sum_{j \in N_0 \cup N_t} p_j\}$ or $\sum_{j \in N'_0} p_j = t$, whichever occurs first. It is possible that the last discard is a fractional job.
3. If $\sum_{j \in N'_t} p_j > T - t$, remove jobs j of N'_t in LPT order by setting $N'_t = N'_t \setminus \{j\}$ until $\sum_{j \in N'_t} p_j = T - t$. It is possible that the last discard is a fractional job.
4. Process the remaining jobs and job pieces of $N'_0 \cup N'_t$ in EDD order.

Step 2 of Algorithm OfflineC attempts to satisfy the constraint that the total size of all scheduled jobs is at most T by discarding the longest jobs of $N'_0 \cup$

N_t'. However, if Step 2 terminates with $\sum_{j \in N_0'} p_j = t$, then Algorithm OfflineC proceeds to Step 3, which discards the longest jobs in N_t' until the total size of all scheduled jobs of N_t is at most $T - t$. In both Steps 2 and 3, the last discard may be a fractional job. The proof of optimality for Algorithm OfflineC is established by a straightforward exchange argument.

For instance I of problem P_C, define $\sigma^O(I)$ to be an optimal offline schedule found by Algorithm OfflineC, $x_j^O(I)$ to be the fraction of job j that is scheduled on-time for $j \in N_0 \cup N_t$, and $z^O(I)$ to be the objective value of this schedule. More generally, when describing an output variable associated with an optimal solution to P_C, we use ^{O} as a superscript.

Finally, we propose online Algorithm BalanceC to solve Problem P_C. This algorithm is equivalent to Algorithm Balance, except that fractions of jobs can be processed and corresponding adjustments are made to Steps 1, 2, and 5. The value of g computed in Step 0 can also be fractional.

Algorithm BalanceC

0. Input $t, T, n, N_0 = E \cup L, p_j$ and d_j for $j \in N_0$, and R^*. Compute $g = n(1 - R^*)$.
1. Partition set E into subsets E_1 and E_2, where $|E_2| = g$, $E_1 = E \setminus E_2$ and $p_i \le p_j$ for all $i \in E_1$ and $j \in E_2$. Set $L' = L \cup E_2$.
2. Partition set L' into subsets L_1' and L_2', where $p_i \le p_j$ for all $i \in L_1'$ and $j \in L_2'$, and $t - p_j < \sum_{i \in E_1 \cup L_1'} p_i \le t$ for all $j \in L_2'$.
   If $\sum_{i \in E_1 \cup L_1'} p_i < t$, then let $k = \text{argmin}_{j \in L_2'} \{p_j\}$. Assign $t - \sum_{i \in E_1 \cup L_1'} p_i$ units of job k to L_1'. Assign the remaining $p_k - (t - \sum_{i \in E_1 \cup L_1'} p_i)$ units of job k to L_2'.
3. Schedule the jobs of $E_1 \cup L_1'$ in EDD order within the interval $[0, t]$.
4. Input N_t, and p_j for $j \in N_t$.
5. Select a maximal set of unscheduled jobs from $(L \cap L_2') \cup N_t$ in SPT order that fit within the interval $[t, T]$, and schedule these jobs in EDD order starting at time t. It is possible that only a fraction of the last job is selected.

In Step 1 of Algorithm BalanceC, since g may be fractional, $|E_1|$ and $|E_2|$ can be fractional. There may be one job k with a fractional part $p_{k_1} \in E_1$ and a fractional part $p_{k_2} = p_k - p_{k_1} \in E_2$. Steps 2 and 5 also allow for fractional jobs. A fractional part k_1 of job k contributes p_{k_1}/p_k towards the number of on-time jobs. Define $\sigma^C(I)$ as a schedule with value $z^C(I)$ found by Algorithm BalanceC for instance I of Problem P_C.

5 Analysis of Algorithm Balance

To establish a lower bound on the competitive ratio of Algorithm Balance, we examine the two cases when $N_t = \emptyset$ and when $N_t \ne \emptyset$. For the first case, we directly show that the lower bound is asymptotically equal to R^*. For the second case, Problem P_C is studied. We show that (e, ℓ)-instances dominate or nearly dominate other instances. More specifically, they can exceed other instances by at most $3/n$. Then, our analysis shows that the performance of Algorithm BalanceC exceeds the performance of Algorithm Balance by at most $2/n$, for large values of n. Hence, the competitive ratio of Algorithm Balance is asymptotically best possible.

We begin by considering the case when $N_t = \emptyset$.

Theorem 8 *If $N_t = \emptyset$, then*

$$\lim_{n\to\infty} R^{\mathcal{B}} = R^*.$$

Proof Suppose $N_t = \emptyset$. From our assumption that all jobs of N_0 satisfy Moore's Algorithm and from Theorem 1, it follows that $z^* = n$. Because

$$z^{\mathcal{B}} \geq n - \lceil n(1 - R^*)\rceil \geq nR^* - 1,$$

$R^{\mathcal{B}} = z^{\mathcal{B}}/z^* \geq R^* - 1/n$. Therefore, $\lim_{n\to\infty} R^{\mathcal{B}} = R^*$. □

We now consider the case when $N_t \neq \emptyset$, and establish some results for an optimal offline schedule.

Lemma 9 *For any instance, there exists an optimal offline schedule σ^* where, for any on-time job $i \in N$ and discarded job $j \in N_0$, $p_i \leq p_j$. Further, if $p_i = p_j$, then $i < j$.*

Proof Consider two jobs i and j in an optimal offline schedule, where $p_i > p_j$ or $i > j$ if $p_i = p_j$, job $i \in N_0$ is on-time and job $j \in N_0$ is discarded. Replace job i with job j in σ^* to create a new schedule. Schedule the jobs of N_0 in EDD order. Then, schedule the jobs of N_t in arbitrary order after those of N_0. Because $p_i \geq p_j$, the new schedule is feasible and hence optimal. A similar replacement argument applied to jobs $i \in N_t$ and $j \in N_0$ in σ^*, where $p_i \geq p_j$, job i is on-time and job j is discarded, shows that there exists an optimal schedule in which job j is on-time and job i is discarded.

This analysis can be repeated for all other job pairs that violate the lemma. □

Remark 10 *There exists an optimal offline schedule where:*

a. $p_j \leq p_i$ *for* $j \in L \cap E^*$ *and* $i \in L \setminus E^*$.
b. *The jobs of $E \cap E^*$ are processed first, followed by the jobs of $L \cap E^*$, followed by the jobs of $L \cap L^*$, followed by the jobs of $N_t \cap S^*$. The jobs in each group are processed in EDD order.*

We henceforth assume that the properties established in Lemma 9 and Remark 10 are satisfied for σ^*. The next two results restrict the jobs of N_t. The first of these shows that there is no advantage to consider instances in which jobs of N_t are discarded by σ^*.

Lemma 11 *In the competitive analysis of Algorithm Balance, it is sufficient to consider instances where all jobs of N_t are on time in some optimal offline schedule σ^*.*

Proof Suppose that $N_t(I) \setminus S^* \neq \emptyset$ for some instance I. Consider instance I' that is identical to instance I, except that $N_t(I') = N_t(I) \cap S^*$. Because no job of $N_t(I) \setminus S^*$ is processed in σ^*, $z^*(I') = z^*(I)$. Since Algorithm Balance has a smaller set of jobs from which to choose in instance I', $z^{\mathcal{B}}(I') \leq z^{\mathcal{B}}(I)$. Thus, $z^{\mathcal{B}}(I')/z^*(I') \leq z^{\mathcal{B}}(I)/z^*(I)$. As a result, instance I is dominated by I' in the competitive analysis of Algorithm Balance. □

The next result restricts our analysis to instances where all jobs of N_t have the same size and fill the available time in σ^*.

Lemma 12 *In the competitive analysis of Algorithm Balance, it is sufficient to consider instances for which $p_j = p^t$, for $j \in N_t$, where*

$$p^t = \min\{T - \sum_{i \in E^* \cup L^*} p_i,\ T - t\}/|N_t|. \tag{10}$$

Proof Suppose instance I satisfies Lemma 11, but does not satisfy the current lemma. Consider a modified instance I' that is identical to I except that the size of each job $j \in N_t(I')$ is $p'_j = \min\{T - \sum_{i \in E^*(I) \cup L^*(I)} p_i,\ T - t\}/|N_t(I)|$. Instance I' has a feasible schedule which is identical to $\sigma^*(I)$, except that the jobs of $N_t(I)$ are replaced by those of $N_t(I')$. Therefore, $z^*(I') = z^*(I)$.

By construction, schedules $\sigma^{\mathcal{B}}(I)$ and $\sigma^{\mathcal{B}}(I')$ are identical in the time interval $[0, t]$. Since $S^{\mathcal{B}}$ contains the shortest jobs of N_t, the inequality $\sum_{j \in S^{\mathcal{B}}(I) \cap N_t(I)} p_j \leq \sum_{j \in S^{\mathcal{B}}(I') \cap N_t(I')} p'_j$ follows. Thus, $z^{\mathcal{B}}(I') \leq z^{\mathcal{B}}(I)$. As a result, $z^{\mathcal{B}}(I')/z^*(I') \leq z^{\mathcal{B}}(I)/z^*(I)$, and instance I' dominates I in the competitive analysis of Algorithm Balance. □

The idle time in $\sigma^{\mathcal{B}}$ complicates our analysis. To resolve this complication, we consider the continuous version of the problem, P_C, as described in Section 4, and the procedure Algorithm BalanceC. We observe that Lemmas 9 – 12 hold for P_C and Algorithm BalanceC. For the remainder of this section, we restrict our attention to instances that satisfy the conditions of these three lemmas.

Remark 13 *For any instance of problem P_C, there exists an optimal offline schedule σ^O where for any on-time job i or on-time fraction of job i, for $i \in N$, and discarded job $j \in N_0$, $p_i \leq p_j$. Further, if $p_i = p_j$, then $i < j$.*

Proof Follows directly from Lemma 9. □

We henceforth assume that an optimal schedule for P_C satisfies Remark 13. The next result shows that if the total size of the jobs discarded by σ^O is no larger than by σ^C, then, in a competitive instance, no job should be discarded by σ^O.

Lemma 14 *In the competitive analysis of Algorithm BalanceC for P_C, it is sufficient to consider instances where $\sum_{j \in E \setminus S^O} p_j > \sum_{j \in E \setminus S^C} p_j$.*

Proof Suppose that some instance I satisfies Lemma 12 and $\sum_{j \in E \setminus S^O} p_j \leq \sum_{j \in E \setminus S^C} p_j$. Because $\sum_{j \in E \setminus S^O} p_j \leq \sum_{j \in E \setminus S^C} p_j$, we have that $|L \cap S^O| \leq |L \cap S^C|$. Also, equation (10) establishes that $N_t \subseteq S^C$. Since $|L \cap S^O| + |N_t| \geq |L|$ and $z^C(I)/z^O(I) < 1$,

$$\begin{aligned}\frac{z^C(I)}{z^O(I)} &= \frac{|E| - g^C + |L \cap S^C| + |N_t|}{|E| - g^O + |L \cap S^O| + |N_t|} \\ &\geq \frac{|E| - g^C + |L|}{|E| - g^O + |L|} \\ &\geq \frac{n - g^C}{n}.\end{aligned}$$

Now consider a similar instance I' where $N_0(I') = N_0(I)$ and $N_t(I') = \emptyset$. Then,

$$\frac{z^C(I')}{z^O(I')} = \frac{n - g^C}{n} \leq \frac{z^C(I)}{z^O(I)},$$

which implies that instance I' dominates instance I. $\square$

Corollary 15 *In the competitive analysis of Algorithm BalanceC for P_C, it is sufficient to consider instances where $g^O > g^C$.*

Proof Since Algorithm BalanceC removes the largest jobs of E, the remark follows directly from Lemma 14. $\square$

Some of the analysis in the current section alters the sizes of various jobs of N_0. Whenever we change these job sizes, we assume that, if needed, the due dates are modified to ensure that the EDD schedule remains feasible. The next result restricts the sizes of the jobs in L.

Lemma 16 *In the competitive analysis of Algorithm BalanceC for P_C, it is sufficient to consider instances where all jobs of $L \cup N_t$ have the same size.*

Proof We first restrict the instances to those where all jobs of $L \cap N_t$ have at most two sizes. Suppose an instance I does not satisfy this condition. Consider an instance I' that is identical to I except that

$$p'_j = \begin{cases} \sum_{i \in S^O \cap L} p_i / |S^O \cap L|, & j \in S^O \cap L \\ \max\{\sum_{i \in S^O \cap L} p_i / |S^O \cap L|, p^t\} & j \in L \setminus S^O \\ p_j, & \text{otherwise,} \end{cases}$$

where p^t is defined in equation (10).

Since the total size of the jobs of $S^O \cap L$ is unchanged and the jobs of $L \setminus S^O$ may replace only jobs of equivalent size in σ^O, $z^O(I') = z^O(I)$. Both σ^0 and σ^C select the shortest jobs of L to process. From Lemma 14, $S^C \cap L \subseteq S^O \cap L$. Thus, $\sum_{i \in S^C \cap L} p'_i \geq \sum_{i \in S^C \cap L} p_i$, and $z^C(I') \leq z^C(I)$. Hence, I' dominates I in the competitive analysis of Algorithm BalanceC. As a result, we assume that $p'_i = \ell = \sum_{i \in S^O \cap L} p_i / |S^O \cap L|$ for all $i \in S^O \cap L$, and $p'_i = \max\{\ell, p^t\}$ for all $i \in L \setminus S^O$.

The remainder of the analysis is divided into two cases: $\ell \leq p^t$, and $\ell > p^t$.
Case 1: $\ell \leq p^t$. Consider an instance I'' that is identical to I' except that $p''_j = \ell$, for $j \in (S^O \setminus L) \cup N_t$. Then,

$$\begin{aligned} \frac{z^C(I')}{z^O(I')} &= \frac{|N_0| - g^C + (T - \sum_{j \in S^C \cap N_0} p'_j)/p^t}{|N_0| - g^O + (T - \sum_{j \in S^O \cap N_0} p'_j)/p^t} \\ &\geq \frac{|N_0| - g^C + (T - \sum_{j \in S^C \cap N_0} p'_j)/\ell}{|N_0| - g^O + (T - \sum_{j \in S^O \cap N_0} p'_j)/\ell} \\ &= \frac{z^C(I'')}{z^O(I'')}. \end{aligned} \tag{11}$$

The inequality follows from the fact that $z^C(I') < z^O(I')$ and $g^C < g^O$ (see Corollary 15) imply that $(T - \sum_{j \in S^C \cap N_0} p'_j)/(T - \sum_{j \in S^O \cap N_0} p'_j) < z^C(I')/z^O(I')$.

Hence, it is sufficient to consider instance I'' that satisfies the conditions of the lemma in the competitive analysis of Algorithm BalanceC.

Case 2: $\ell > p^t$. Then, $N_t(I') \subseteq L^C(I')$ and $N_t(I') \subseteq L^O(I')$. Thus, increasing the total size of the jobs in N_t by $\ell - p^t > 0$, reduces z^C by the same amount as z^O. Thus, instance I', which dominates instance I, is itself dominated in the competitive analysis of Algorithm BalanceC. □

From Lemma 16, we let ℓ denote the common size of the jobs of $L \cup N_t$ for the instances that we consider in our competitive analysis of Algorithm BalanceC.

The next result establishes restrictions on a set of instances that increase the competitive ratio by a relatively small amount.

Lemma 17 *Restricting instances to those where $\sum_{j \in E} p_j = t$ and $p_j = e = t/|E|$ for $j \in E$, increases the competitive ratio of Algorithm BalanceC for P_C by less than $3/n$, for $n \geq 3$.*

Proof Consider an instance I. The mean size of the jobs discarded from $E(I)$ in σ^O is

$$\bar{e} = \frac{t - \sum_{j \in E \cap S^O} \hat{p}_j}{|E \setminus S^O|}, \tag{12}$$

where $\hat{p}_j =$ the amount of job $j \in E$ that is processed by σ^O in the interval $[0, t]$. Observe that $|E \setminus S^O|$ is fractional if $0 < \hat{p}_j < p_j$ for some $j \in E$.

Construct the following instance I' from I. Let $E(I')$ contain $\lceil t/\bar{e} \rceil$ identical jobs j, each with size $p'_j = t/\lceil t/\bar{e} \rceil$ and due date $d'_j = t$. Also, let $L(I')$ contain $\lfloor |E(I)| - |E(I')| \rfloor$ jobs j, each with size $p'_j = 0$ and due date $d'_j = T$, and the jobs of set L. If a fraction of some job $r \in N_0$ is processed by σ in $[0, t]$, then we also construct a job with size $\sum_{j \in E} p_j - t$ and due date T. Finally, let $N_t(I') = N_t(I)$. As a result, $n(I') = n(I)$.

We now construct a feasible schedule, $\sigma'(I)$, for instance I' based on $\sigma^O(I)$. Discard $(t - \sum_{j \in E \cap S^O} \hat{p}_j)/[t/\lceil t/\bar{e} \rceil]$ jobs of $E(I')$. As a result, $\sigma'(I')$ discards the same total size of early jobs as $\sigma^O(I)$. Moreover, the number of jobs discarded is

$$\begin{aligned}
\frac{[t - \sum_{j \in E \cap S^O} \hat{p}_j)] \lceil t/\bar{e} \rceil}{t} &< \frac{[t - \sum_{j \in E \cap S^O} \hat{p}_j)](t/\bar{e} + 1)}{t} \\
&= |E \setminus S^O| + \frac{t - \sum_{j \in E \cap S^O} \hat{p}_j}{t} \\
&< |E \setminus S^O| + 1,
\end{aligned}$$

where the equality follows from equation (12). Thus, less than one more job is discarded in $\sigma'(I')$ than in $\sigma^O(I)$. In $\sigma'(I')$, the jobs $j \in L(I')$ with $p'_j = 0$ are on time, the job constructed to replace r is on time, and each job $j \in L \cup N_t$ is on time. Since $\sigma'(I')$ is feasible, $z^O(I') \geq z^O(I) - 1$.

Algorithm OfflineC discards the longest jobs of E, each of which has size greater than ℓ. From Lemma 14, $E \setminus S^C \subset E \setminus S^O$. As a result of (12), the average job size removed by Algorithm BalanceC is no larger for instance I' than for instance I. Thus, Algorithm BalanceC has no additional time to schedule the job corresponding to r and the jobs of $L(I') \cup N_t(I')$ in instance I' than it does to schedule the jobs of $\{r\} \cup L(I) \cup N_t(I)$ in instance I. Note that r is a fractional job in instance I, but the corresponding job in I' is a complete job. Therefore, $z^C(I') < z^C(I) + 1$. Hence,

$$\frac{z^C(I')}{z^O(I')} < \frac{z^C(I)+1}{z^O(I)-1} \leq \frac{z^C(I)}{z^O(I)} + \frac{3}{n},$$

where the second inequality follows from $n \geq 3$, which implies $z^O(I) \geq 3$. □

The next result restricts the total size of the jobs of L.

Lemma 18 *In the competitive analysis of Algorithm BalanceC for P_C, it is sufficient to consider instances where $p_j = \ell$ for $j \in L \cup N_t$, and where $\sum_{j\in L} p_j = \min\{T-t,\ t\}$.*

Proof Consider an instance I, where we assume that $p_j = \ell$ for $j \in L \cup N_t$ from Lemma 16.

If $\sum_{j\in L(I)} p_j > \min\{T-t,\ t\}$, then analysis similar to that used in (11) shows that decreasing the size of ℓ also reduces the value of z^C/z^O. Hence, I is dominated by an instance where $\sum_{j\in L(I)} p_j = \min\{T-t,\ t\}$.

Alternatively, suppose $\sum_{j\in L(I)} p_j < \min\{T-t,\ t\}$. Create an instance I' by adding a suitably small fraction $\epsilon > 0$ of a job $n+1$ to L, where $p_{n+1} = \ell$ and $d_{n+1} = T$. Because $e > \ell$, this fraction of job $n+1$ replaces a portion of the jobs of E in optimal schedule σ^O. Thus, $z^O(I') = z^O(I) + \epsilon(e-\ell)/e$.

Because, $g^C(I) = n(1-R^*)$, $g^C(I') = g^C(I) + \epsilon(1-R^*)$. Consequently, $z^C(I') = z^C(I) + \epsilon(1-R^*)(e-\ell)/e$. As a result,

$$\frac{z^C(I')}{z^O(I')} = \frac{z^C(I) + (1-R^*)\epsilon(e-\ell)/e}{z^O(I) + \epsilon(e-\ell)/e} < \frac{z^C(I)}{z^O(I)},$$

where the inequality follows from Remark 2. Thus, instance I' dominates I. □

We now return to the analysis of Algorithm Balance. The next result restricts the analysis to the (e,ℓ)-instances considered in Section 3.

Lemma 19 *Restricting instances to those where $\sum_{j\in E} p_j = t$, $p_j = e$ for $j \in E$, $p_j = \ell$ for $j \in L \cup N_t$, where $e > \ell$, and $\sum_{j\in L} p_j = \min\{T-t,\ t\}$, increases the competitive ratio of Algorithm Balance by less than $5/n$, for $n \geq 3$.*

Proof Consider a competitive instance I for Algorithm Balance. A solution to P_C using Algorithm BalanceC has at most two fractional jobs. Thus, $z^{\mathcal{B}}(I) \geq z^C(I) - 2$. Also, $z^*(I) \leq z^O(I)$. Generate instance I' that satisfies the properties established in Lemmas 16 – 17 using the constructive approaches that appear in the proofs of these lemmas. Thus,

$$\begin{aligned} \frac{z^{\mathcal{B}}(I)}{z^*(I)} &\geq \frac{z^C(I)-2}{z^O(I)} \\ &\geq \frac{z^C(I)}{z^O(I)} - \frac{2}{n} \\ &\geq \left(\frac{z^C(I')}{z^O(I')} - \frac{3}{n}\right) - \frac{2}{n}, \end{aligned}$$

where the last inequality follows from Lemma 17. □

For the remainder of this section, we assume that an instance satisfies the conditions of Lemma 19. The analysis separately considers the two cases: $T/t < 2$ and $T/t \geq 2$.

Lemma 20 *If $T/t < 2$ and $n \geq 3$, then*

$$R^{\mathcal{B}}(n) > \frac{(1+\alpha)^2}{(1+\alpha)^2+\alpha^2} - \frac{5}{n}. \tag{13}$$

Proof From Lemma 19, consider the following problem that overestimates the bound for $R^{\mathcal{B}}(n)$ by less than $5/n$:

$$\min_{e,\ell} R(n) = \frac{t/e - g^{\mathcal{B}} + [T-(t-g^{\mathcal{B}}e)]/\ell}{(2t-T)/e + 2(T-t)/\ell} \tag{14}$$

$$\text{s.t.} \qquad t/e + (T-t)/\ell = n \tag{15}$$

$$e > \ell > 0. \tag{16}$$

Since $e > \ell$, all jobs of L replace jobs of E in σ^*. In $\sigma^{\mathcal{B}}$, $g^{\mathcal{B}}$ jobs of E are replaced by jobs of L. These substitutions give (14). From (14) – (16),

$$\begin{aligned} R(n) &= \frac{t/e - g^{\mathcal{B}} + [T-(t-g^{\mathcal{B}}e)]/\ell}{(2t-T)/e + 2(T-t)/\ell} \\ &= \frac{t\ell - g^{\mathcal{B}}e\ell + Te - te + g^{\mathcal{B}}e^2}{2Te - 2te + 2t\ell - T\ell} \\ &= \frac{t(T-t)e - g^{\mathcal{B}}(T-t)e^2 + T(ne-t)e - t(ne-t)e + g^{\mathcal{B}}(ne-t)e^2}{2T(ne-t)e - 2t(ne-t)e + 2t(T-t)e - T(T-t)e} \\ &= \frac{ng^{\mathcal{B}}e^2 - Tg^{\mathcal{B}}e + n(T-t)e}{2n(T-t)e - T^2 + Tt}, \end{aligned} \tag{17}$$

where the third equality follows from (15). The first order condition from (17) is

$$\begin{aligned} \frac{dR(n)}{de} &= 0 = (2ne-T)(T-t)(2ng^{\mathcal{B}}e - Tg^{\mathcal{B}} + nT - nt) \\ &\qquad -2n(T-t)(ng^{\mathcal{B}}e^2 - Tg^{\mathcal{B}}e + nTe - nte) \\ &\quad = 2n^2g^{\mathcal{B}}e^2 - 2nTg^{\mathcal{B}}e + (T^2g^{\mathcal{B}} - nT^2 + nTt). \end{aligned}$$

Thus,

$$ne = \frac{Tg^{\mathcal{B}} \pm T\sqrt{2n\alpha^2 g^{\mathcal{B}} - (g^{\mathcal{B}})^2}}{2g^{\mathcal{B}}}.$$

First, suppose $ne = [Tg^{\mathcal{B}} - T\sqrt{2n\alpha^2 g^{\mathcal{B}} - (g^{\mathcal{B}})^2}]/2g^{\mathcal{B}}$. Since $ne < T/2$, (15) is violated. Hence, a minimum feasible value of $R(n)$ is not attained.

Alternatively, suppose $ne = [Tg^{\mathcal{B}} + T\sqrt{2n\alpha^2 g^{\mathcal{B}} - (g^{\mathcal{B}})^2}]/2g^{\mathcal{B}}$. Then, $R(n)$ is a decreasing function of ne for $T/2 < ne < [Tg^{\mathcal{B}} + T\sqrt{2n\alpha^2 g^{\mathcal{B}} - (g^{\mathcal{B}})^2}]/2g^{\mathcal{B}}$, and an increasing function of ne for $ne > [Tg^{\mathcal{B}} + T\sqrt{2n\alpha^2 g^{\mathcal{B}} - (g^{\mathcal{B}})^2}]/2g^{\mathcal{B}}$. Consequently, $R(n)$ is minimized when $ne = [Tg^{\mathcal{B}} + T\sqrt{2n\alpha^2 g^{\mathcal{B}} - (g^{\mathcal{B}})^2}]/2g^{\mathcal{B}}$. Then from (17),

$$\begin{aligned}R(n) &= \frac{[Tg^{\mathcal{B}}+T\sqrt{2n\alpha^2 g^{\mathcal{B}}-(g^{\mathcal{B}})^2}]^2/4ng^{\mathcal{B}}+(nT\alpha^2-Tg^{\mathcal{B}})(Tg^{\mathcal{B}}+T\sqrt{2n\alpha^2g^{\mathcal{B}}-(g^{\mathcal{B}})^2})/2ng^{\mathcal{B}}}{T\alpha^2(Tg^{\mathcal{B}}+T\sqrt{2n\alpha^2g^{\mathcal{B}}-(g^{\mathcal{B}})^2})/g^{\mathcal{B}}-T^2\alpha^2}\\
&= \frac{(Tg^{\mathcal{B}}+T\sqrt{2ng^{\mathcal{B}}\alpha^2-(g^{\mathcal{B}})^2})(nT\alpha^2+T\sqrt{2n\alpha^2g^{\mathcal{B}}-(g^{\mathcal{B}})^2}/2-Tg^{\mathcal{B}}/2)}{2nT^2\alpha^2\sqrt{2n\alpha^2g^{\mathcal{B}}-(g^{\mathcal{B}})^2}}\\
&= \frac{n\alpha^2+\sqrt{2n\alpha^2g^{\mathcal{B}}-(g^{\mathcal{B}})^2}}{2n\alpha^2}.\end{aligned}$$

Since $g^{\mathcal{B}} = \lceil g_0 \rceil = \lceil n\alpha^2/[(1+\alpha)^2+\alpha^2] \rceil < 2n\alpha^2$,

$$\begin{aligned}R(n) &\geq \frac{n\alpha^2+\sqrt{2n\alpha^2 g_0-(g_0)^2}}{2n\alpha^2}\\
&= \frac{n\alpha^2+\sqrt{2n\alpha^2\frac{n\alpha^2}{(1+\alpha)^2+\alpha^2}-\left(\frac{n\alpha^2}{(1+\alpha)^2+\alpha^2}\right)^2}}{2n\alpha^2}\\
&= \frac{(1+\alpha)^2}{(1+\alpha)^2+\alpha^2}\\
&= R^*.\end{aligned}$$

The statement of the lemma now follows from $R^{\mathcal{B}} \leq R(n) + 5/n$. □

Lemma 21 *If $T/t \geq 2$ and $n \geq 3$, then*

$$R^{\mathcal{B}}(n) > \frac{(1+\gamma)^2}{(1+\gamma)^2+1} - \frac{5}{n}. \tag{18}$$

Proof From Lemma 19, consider the following problem that overestimates the bound for $R^{\mathcal{B}}(n)$ by less than $5/n$:

$$\min_{e,\ell} R(n) = \frac{t/e - g^{\mathcal{B}} + [T-(t-g^{\mathcal{B}}e)]/\ell}{T/\ell} \tag{19}$$

$$\text{s.t.} \qquad t/e + t/\ell = n \tag{20}$$

$$e > \ell > 0. \tag{21}$$

Since $e > \ell$, all jobs of E are replaced by jobs of L in σ^*. In $\sigma^{\mathcal{B}}$, $g = g^{\mathcal{B}}$ jobs of E are replaced by jobs of L. These substitutions give (19). From (19) – (21),

$$\begin{aligned}R(n) &= \frac{t/e-g^{\mathcal{B}}+[T-(t-g^{\mathcal{B}}e)]/\ell}{T/\ell}\\
&= \frac{t\ell-g^{\mathcal{B}}e\ell+Te-te+g^{\mathcal{B}}e^2}{Te}\\
&= \frac{t^2e-tg^{\mathcal{B}}e^2+(ne-t)(Te-te+g^{\mathcal{B}}e^2)}{T(ne-t)e}\\
&= \frac{2t^2-2tg^{\mathcal{B}}e+nTe-Tt-nte+ng^{\mathcal{B}}e^2}{T(ne-t)},\end{aligned} \tag{22}$$

where the third equality follows from (20). The first order condition from (22) is

$$\begin{aligned}\frac{dR(n)}{db} &= 0 = (ne-t)(-2g^{\mathcal{B}}t+nT-nt+2ng^{\mathcal{B}}e)-n(2t^2-2tg^{\mathcal{B}}e+nTe-Tt-nte+ng^{\mathcal{B}}e^2)\\ &= n^2g^{\mathcal{B}}e^2-2ntg^{\mathcal{B}}e+(2t^2g^{\mathcal{B}}-nt^2).\end{aligned}$$

Thus,

$$ne = \frac{Tg^{\mathcal{B}} \pm \sqrt{T^2(n-g^{\mathcal{B}})g^{\mathcal{B}}}}{\gamma^2 g^{\mathcal{B}}}.$$

First, suppose $ne = [Tg^{\mathcal{B}} - \sqrt{T^2(n-g^{\mathcal{B}})g^{\mathcal{B}}}]/\gamma^2 g^{\mathcal{B}}$. Since $ne < T/\gamma^2 = t$, (20) is violated. Hence, a minimum feasible value of $R(n)$ is not attained.

Alternatively, suppose $ne = [Tg^{\mathcal{B}} + \sqrt{T^2(n-g^{\mathcal{B}})g^{\mathcal{B}}}]/\gamma^2 g^{\mathcal{B}}$. Then, $R(n)$ is a decreasing function of ne for $T/\gamma^2 < ne < [g^{\mathcal{B}}T + \sqrt{T^2(n-g^{\mathcal{B}})g^{\mathcal{B}}}]/\gamma^2 g^{\mathcal{B}}$, and an increasing function of ne for $ne > [Tg^{\mathcal{B}} + \sqrt{T^2(n-g^{\mathcal{B}})g^{\mathcal{B}}}]/\gamma^2 g^{\mathcal{B}}$. Consequently, $R(n)$ is minimized when $ne = [Tg^{\mathcal{B}} + \sqrt{T^2(n-g^{\mathcal{B}})g^{\mathcal{B}}}]/\gamma^2 g^{\mathcal{B}}$. Then from (22),

$$\begin{aligned}R(n) &= \frac{[2t^2-Tt+(T-t)ne-2tg^{\mathcal{B}}e+ng^{\mathcal{B}}e^2]\gamma^2 g^{\mathcal{B}}}{T\sqrt{T^2(n-g^{\mathcal{B}})g^{\mathcal{B}}}}\\ &= \frac{[2T^2/\gamma^4-T^2/\gamma^2+[Tg^{\mathcal{B}}+\sqrt{g^{\mathcal{B}}T^2(n-g^{\mathcal{B}})}][\gamma^2nT-nT-Tg^{\mathcal{B}}+\sqrt{g^{\mathcal{B}}T^2(n-g^{\mathcal{B}})}]/n\gamma^4g^{\mathcal{B}}]\gamma^2g^{\mathcal{B}}}{T\sqrt{T^2(n-g^{\mathcal{B}})g^{\mathcal{B}}}}\\ &= \frac{2T^2g^{\mathcal{B}}-T^2\gamma^2g^{\mathcal{B}}+[Tg^{\mathcal{B}}+\sqrt{g^{\mathcal{B}}T^2(n-g^{\mathcal{B}})}][\gamma^2nT-nT-Tg^{\mathcal{B}}+\sqrt{T^2(n-g^{\mathcal{B}})g^{\mathcal{B}}}]/n}{T\gamma^2\sqrt{T^2(n-g^{\mathcal{B}})g^{\mathcal{B}}}}\\ &= \frac{n(\gamma^2-1)+2\sqrt{g^{\mathcal{B}}(n-g^{\mathcal{B}})}}{n\gamma^2}.\end{aligned}$$

Remark 2 shows that $n - g_0 > n/2$. Since $g^{\mathcal{B}} = \lceil g_0 \rceil \geq g_0$,

$$\begin{aligned}R(n) &\geq \frac{n(\gamma^2-1)+2\sqrt{g_0(n-g_0)}}{n\gamma^2}\\ &= \frac{(1+\gamma)^2}{(1+\gamma)^2+1}\\ &= R^*.\end{aligned}$$

The statement of the lemma now follows from $R^{\mathcal{B}} \leq R(n) + 5/n$. □

A summary of the asymptotic results for Algorithm Balance, which is the main result of this section, is now provided.

Theorem 22

$$\lim_{n\to\infty} R^{\mathcal{B}}(n) = \begin{cases} \dfrac{(1+\alpha)^2}{(1+\alpha)^2+\alpha^2}, & T/t < 2 \qquad (23)\\[2ex] \dfrac{(1+\gamma)^2}{(1+\gamma)^2+1}, & T/t \geq 2. \qquad (24)\\ & \qquad (25)\end{cases}$$

Proof When $N_t = \emptyset$, Theorem 8 establishes both equations (23) and (24).

When $N_t \neq \emptyset$ and $T/t < 2$, Theorem 7 and Lemma 20 show that $R^* - 5/n < R^{\mathcal{B}} < R^* + 1/n$. This establishes (23). When $N_t \neq \emptyset$ and $T/t \geq 2$, Theorem 7 and Lemma 21 show that $R^* - 5/n < R^{\mathcal{B}} < R^* + 1/n$. This establishes (24). $\square$

The results established in Theorems 7 and 22 show that the performance of Algorithm Balance is asymptotically best possible.

6 Concluding Remarks

In this paper, we consider a two-period planning environment. In the first planning period, complete information about available jobs is known. However, the possibility that unspecified jobs will arrive at the start of the second period complicates decision making in the first period. Our contribution is the design and analysis of an online algorithm that asymptotically achieves the best possible competitive ratio.

The problem studied here has various features in common with production planning, planning horizon, and online scheduling problems. Similarities with production planning include the unpredictable arrival of future orders, and the need to reject some orders due to limited capacity (Geunes et al., 2005). There are also differences. Production planning problems with stochastic future demand typically use minimization of expected cost as an objective (Ciarallo et al., 1994). However, since we have no distributional information, we consider the worst case performance relative to that of an offline algorithm with complete future knowledge. We also assume that future orders can only arrive at one known time, rather than many possible times. Similarities with planning horizon problems include unknown future demand (Chand et al., 2002) and evaluation of an algorithm that has available information for only part of the planning horizon (Modigliani and Hohn, 1955). Similarities with online scheduling include arbitrary future demand and order rejection, and the use of competitive ratio as a performance measure (Pruhs et al., 2004). However, we assume knowledge of a current order set, as is relevant in many practical situations. Our work also extends the literature of online planning period scheduling problems by considering a different objective (Hall et al., 2009).

Several related research opportunities are available for future study. First, our model can be generalized to consider arbitrary due dates for jobs that arrive at the start of the second period. Second, a variety of objectives other than maximizing the number of on-time jobs can be considered. Finally, the new topic of online planning period problems is rich with additional research questions that also have practical relevance. In conclusion, we hope that our work will encourage further investigation into these interesting research issues.

Acknowledgements This research is supported in part by National Science Foundation Grant DMI-0421823, and by the Summer Fellowship Program of the Fisher College of Business, The Ohio State University, to the first author.

References

P. Baptiste, An $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Operations Research Letters* **24** (1999) 175–180.

J.C. Bean, J.R. Birge, J. Mittenthal, C.E. Noon, Matchup scheduling with multiple resources, release dates and disruptions. *Operations Research* **39** (1991) 470–483.

M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest, R.E. Tarjan, Time bounds for selection. *Journal of Computer and System Sciences* **7** (1972) 448–461.

S. Chand, V.N. Hsu, S. Sethi, Forecast, solution and rolling horizons in operations management problems: A classified bibliography. *Manufacturing & Service Operations Management* **4** (2002) 25–43.

M. Chrobak, M. Jawor, J. Sgall, T. Tichy, Online scheduling of equal-length jobs: Randomization and restarts help. *SIAM Journal on Computing* **36** (2006) 1709–1728.

F.W. Ciarallo, R. Akella, T.E. Morton, A periodic review, production planning model with uncertain capacity and uncertain demand - Optimality of extended myopic policies. *Management Science* **40**(3) (1994) 320–332.

S.P.Y. Fung, Lower bounds on online deadline scheduling with preemption penalties. *Information Processing Letters* **108** (2008) 214–218.

J. Geunes, Y. Merzifonluoğlu, H.E. Romeijn. K. Taaffe, Demand selection and assignment problems in supply chain planning. *Tutorials in Operations Research*, INFORMS (2005) 124–141.

R.L. Graham, Bounds for certain multiprocessor anomalies. *Bell System Technical Journal* **45** (1966) 1563–1581.

N.G. Hall, M.E. Posner, C.N. Potts, Online scheduling with known arrival times. *Mathematics of Operations Research* **34** (2009) 92–102.

H. Hoogeveen, C.N. Potts, G.J. Woeginger, On-line scheduling on a single machine: Maximizing the number of early jobs. *Operations Research Letters* **27** (2000) 193–197.

E. Ignall, A.F. Veinott, Jr., Optimality of myopic inventory policies for several substitute products. *Management Science* **16** (1969) 284–304.

J.R. Jackson, Scheduling a production line to minimize maximum tardiness. Research Report 43, Management Science Research Project, University of California, Los Angeles, CA, 1955.

E.L. Lawler, Scheduling a single machine to minimize the number of late jobs. Technical report, University of California at Berkeley, Berkeley, CA, 1983.

J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems. *Annals of Operations Research* **1** (1977) 343–362.

C. Li, F. Zhang, Advance demand information, price discrimination, and preorder strategies. *Manufacturing & Service Operations Management* **15**(1) (2013) 57–71.

J. McClain, L.J. Thomas, Horizon effects in aggregate production planning with seasonal demands. *Management Science* **23** (1977) 728–736.

F. Modigliani, F.E. Hohn, Production planning over time and the nature of the expectation and planning horizon. *Econometrica* **23** (1955) 46–66.

J.M. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science* **15** (1968) 102–109.

M.L. Pinedo, 2012. *Scheduling: Theory, Algorithms and Systems*, fourth ed., Springer, 2012.

K. Pruhs, J. Sgall, E. Torng. Online scheduling,in: J.Y.-T. Leung (Ed.), *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, CRC Press, Boca Raton, FL, 2004.

R.D. Reid, N.R. Sanders, *Operations Management: an Integrated Approach*, second ed., John Wiley, Hoboken, NJ, 2005.

R.S. Russell, B.W. Taylor, III, *Operations Management: Quality and Competitiveness in a Global Environment*, fifth ed., John Wiley, Hoboken, NJ, 2006.

W.E. Smith, Various optimizers for single stage production. *Naval Research Logistics Quarterly* **3** (1956) 59–66.

X. Zhao, K.E. Stecke, Pre-orders for new to-be-released products considering consumer loss aversion. *Production and Operations Management* **19**(2) (2010) 198–215.

F.F. Zheng, Y.F. Xu, E. Zhang, On-line production order scheduling with production order penalties. *Journal of Combinatorial Optimization* **13** (2007) 189–204.

F.F. Zheng, Y.F. Xu, E. Zhang, How much can lookahead help in online single machine scheduling. *Information Processing Letters* **106** (2008) 70–74.