ELSEVIER

Operations Research Letters 29 (2001) 141–148

operations research letters

www.elsevier.com/locate/dsw

Asymptotic analysis of an on-line algorithm for the single machine completion time problem with release dates ☆

Philip Kaminsky[a, *], David Simchi-Levi[b]

[a] *Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720-1777, USA*
[b] *The Engineering Systems Division and the Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA*

Received 11 October 1997; received in revised form 1 May 2001; accepted 10 July 2001

Abstract

In the single machine mean completion time problem with release dates, a set of jobs has to be processed non-preemptively on a single machine. No job can be processed before its release date, and the objective is to determine a sequence of the jobs on the machine which minimizes the sum of the completion times of all jobs. In this paper, we prove the asymptotic optimality of the shortest processing time among available jobs algorithm, in which at the completion time of any job, the next job to be scheduled is the shortest job among all those released but not yet processed. This algorithm is particularly attractive because it falls in the class of easy to implement and computationally inexpensive *on-line* algorithms. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Scheduling; Asymptotic analysis; On-line algorithms

0. Introduction

In the *single machine mean completion time problem with release dates*, a set of jobs must be processed non-preemptively on a single machine. The machine can only process one job at a time. Each job has two associated parameters: a processing time, and a release time. The objective is to determine a sequence of the jobs on the machine such that no job can be processed before its release time and so as to minimize the average, or equivalently the sum, of the completion time of the jobs.

It is well known that the version of this model *without release dates*, i.e., when all jobs are available for processing at the same time, is trivially solved using the shortest processing time first sequence, see [14]. On the other hand, the problem with release date was shown to be NP-Complete, see [12], which motivated researchers to develop branch and bound based algorithms for the problem. This includes Bianco and Ricciardelli [3], Dessouky and Deogun [5], Hariri and Potts [8], and Beloudadah et al. [2].

More recently, cutting plane based algorithms have also been developed. For instance, Dyer and Wolsey [6] developed valid inequalities for linear programming formulations of various relaxations of this problem. Sousa and Wolsey [16] investigated a time-indexed integer programming formulation of this model, developed a set of valid inequalities, and implemented a branch and cut algorithm based on these

☆ Research supported in part by ONR Contracts N00014-95-1-0232, and N00014-01-1-0146, NSF Contracts DDM-9322828 and DMI-9732795, and a grant from S&C Electric Corporation.

* Corresponding author.
E-mail address: kaminsky@ieor.berkeley.edu (P. Kaminsky).

PII: S0167-6377(01)00099-2

inequalities. Finally, Van der Akker [17] developed a branch and cut algorithm for the model.

Due to the complexity of the problem, it is of no surprise that heuristics algorithms have also been developed, analyzed and tested. Indeed, recently performance guarantees have been established for various *on-* and *off-line* heuristics. In this context, on-line scheduling algorithms sequence jobs at any time using only information pertaining to jobs which have been released by that time. In contrast, off-line algorithms may use information about jobs which will be released in the future.

For example, Phillips et al. [13], and Hoogeveen and Vestjens [9] developed heuristic algorithms with worst-case bounds of two. This implies that for any instance of the machine scheduling problem, the ratio of the objective value produced by this algorithm to the objective value of the optimal solution is no more than two. Interestingly, Hoogeveen and Vestjens [9] also prove that *no* on-line algorithm can have a worst-case bound better than two. Chekuri et al. [4] introduced an off-line algorithm whose worst-case bound is approximately 1.58. Recently, Afrati et al. [1] developed a polynomial time approximation scheme for this problem.

Probabilistic analysis has been applied, by Gazmuri [7], to the single machine mean completion time model with release dates. In his work, however, Gazmuri distinguishes between two cases. In the first, the expected processing time is assumed to be less than the expected inter-arrival time while in the second case the expected processing time is assumed to be greater than the expected inter-arrival time. In each case, he develops a different algorithm; in the first case, an off-line algorithm while in the second case, an on-line algorithm. In both cases it is shown that the relative error between the solution generated by each algorithm and the optimal solution decreases to zero as the number of jobs increases. Unfortunately, the case when expected processing time equals expected interarrival time remains open.

In contrast, in this paper, we prove that a simple on-line dispatch rule, shortest processing time among available jobs (SPTA), is asymptotically optimal for the single machine completion time problem with release dates, *regardless* of the relationship between expected inter-arrival time and expected processing time. In fact, this result follows as a simple corollary of the main result of this paper, which relates the completion time of jobs in the SPTA sequencing to the completion time of jobs in a simple lower bound.

In the next section, we formally state our main result, and its accompanying corollary. This is followed by a series of properties and lemmas which culminates with the proof of the main result.

1. The model and the main results

Consider a set of n jobs that have to be processed on a single machine. Job i, $i = 1, 2, \ldots, n$, has a processing time p_i, and a release time r_i. Each job must be processed without preemption on the machine. Each job is available for processing at its release time, and may not be processed before that time. The objective is to determine a *schedule*, or sequence of jobs, such that the total completion times of all of the jobs on the machine is minimized. We call this problem Problem P and use Z^* to denote its optimal objective function value. That is, Z^* is the minimum possible total completion time of all jobs in Problem P. Similarly, given a heuristic H for the single machine completion time problem with release dates, we use Z^H to denote the sum of completion times in the resulting schedule.

Specifically, we consider the following heuristic: At the completion time of any job, consider all the jobs which have been released by that time, and select the job with the smallest processing time to be processed next. If no job is available, the machine is idle until at least one job arrives. We call this heuristic the SPTA heuristic, and denote the sum of completion times of all of the jobs when they are scheduled using this heuristic Z^{SPTA}.

To analyze the effectiveness of the SPTA heuristic, we relate its performance to that of a lower bound on the optimal solution value. To construct the lower bound, we relax the original problem statement to allow for preemption. Clearly, the optimal solution to the relaxed problem provides a lower bound on the original problem. However, unlike the original problem, the optimal solution for the relaxed problem can be found in polynomial time, see [15], using the shortest remaining processing time (SPTR) strategy. In this algorithm, whenever a new job is *released*, we compare its processing time to the *remaining processing time* of the job currently being processed, as well as

the *remaining processing time* of any jobs which have previously been preempted. The shortest among these is selected to be processed next. Similarly, whenever a job completes processing, we select the job with the shortest *remaining processing time* for processing. Without loss of generality, we assume that whenever we are indifferent between several jobs, i.e., they have equal remaining processing times, we select the job which originally appeared first in the lower bound. We denote the sum of the completion times of all jobs scheduled using this strategy Z^{SPTR}, and note that since this is a relaxation of Problem P, we have that

$$Z^{\mathrm{SPTR}} \leqslant Z^* \leqslant Z^{\mathrm{SPTA}}.$$

To relate the upper bound to the lower bound, consider an instance of Problem P, and let $C_i,\ i = 1, 2, \ldots, n$, be the completion time of the ith job to complete in a solution generated by the SPTA heuristic. Similarly, let $\underline{C}_i,\ i = 1, 2, \ldots, n$, be the completion time of the ith job *to complete* in a sequence generated by the SPTR strategy. Note that these are not necessarily the same jobs. Also note that in an SPTR strategy, the ith job to complete processing is not necessarily the ith job to begin processing. Let $\bar{P}$ equal the largest processing time among all jobs in Problem P.

We bound the difference between the lower and upper bounds in the following theorem:

Theorem 1.1. *Consider Problem P, the single machine completion time problem with release dates. For any i,* $i = 1, 2, \ldots, n$, *we have*

$$C_i - \underline{C}_i \leqslant \bar{P}.$$

The intuition behind this relationship lies in the fact that the non-preemptive version of this heuristic can never fall too far behind the preemptive version. Intuitively, the reason that the SPTA heuristic performs worse than the SPTR heuristic is that SPTA might start processing a long job, and a series of short jobs may arrive immediately after the long job starts processing. Of course, SPTR will preempt the "long job" in order to process the series of short jobs. However, the SPTA heuristic will have an opportunity to "catch up" as soon as the currently processing jobs are complete. Hence, there is no reason for these jobs to complete more than the processing time of the "long job" later in the upper bound than in the lower bound, unless even shorter jobs arrive.

This theorem indicates that for the average completion time objective, the relative error for the SPTA solution does not get too large. The following corollary is a direct consequence of Theorem 1.1.

Corollary 1.2. *Let the processing times* $p_i,\ i = 1, 2, \ldots, n$, *be defined on a bounded interval. Then*

$$\lim_{n\to\infty} \frac{Z^{\mathrm{SPTR}}}{n^2} = \lim_{n\to\infty} \frac{Z^*}{n^2} = \lim_{n\to\infty} \frac{Z^{\mathrm{SPTA}}}{n^2}.$$

Interestingly, the corollary strengthens, in an asymptotic sense, a result of Checkuri et al. [4], who prove that for any instance of Problem P,

$$Z^* \leqslant \left(\frac{e}{e-1}\right) Z^{\mathrm{SPTR}}.$$

2. Proof of the main theorem

To prove Theorem 1.1, we need a series of properties, which are presented below. In what follows, when we refer, in the context of the lower bound, to the *remaining processing time* of job j at time t, we are referring to the total length of job j minus that portion which has already been processed in the lower bound by time t. Also, when we refer to the interval of time between x and y, we are referring to the interval $(x, y]$.

Observe that both the lower and upper bounds consist of series of contiguous jobs or portions of jobs, occasionally interrupted by idle time. We call each series of contiguous jobs or portions of jobs a *chain*. Number the chains separately and consecutively in the upper and lower bounds, and label chain i in the upper bound H_i, and in the lower bound $\underline{H}_i$. We present the following property without proof:

Property 2.1. *For n jobs, there will be exactly the same number h of chains in the upper and lower bounds, and for each i,* $i = 1, 2, \ldots, h$, *chains* H_i *and* $\underline{H}_i$ *contain exactly the same jobs, and start and end at exactly the same time.*

For the following properties, consider a single chain in the upper bound, and its corresponding chain in the lower bound. Let $B_i,\ i = 1, 2, \ldots, n$ represent the

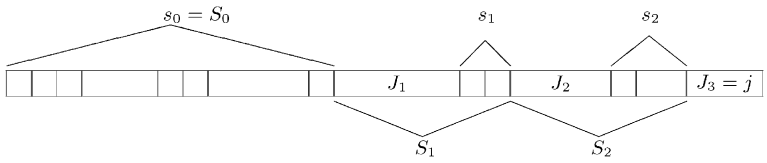

Fig. 1. Marked jobs.

time the ith job *begins* processing in the upper bound. From the definition of the SPTA heuristic, it is clear that job i is the shortest job with release time less than or equal to B_i, which has not already appeared in the upper bound sequence by time B_i.

Next, consecutively number the jobs in the upper bound, and consider job j. We call all jobs $i \leqslant j$ such that $p_i \geqslant \max_{i \leqslant k \leqslant j} p_k$ *marked*. Note that different jobs may be *marked*, depending on the choice of j. We number the marked jobs from 1 to m_j and label them J_i, $i = 1, 2, \ldots, m_j$, and note that $J_{m_j} = j$. To simplify notation we let J_0 be a dummy job of size 0 representing the beginning of the chain and refer to that dummy job as a marked job. Let s_i, $i = 0, 1, 2, \ldots, m_j$ represent the set of jobs, possibly empty, which appear in the upper bound between consecutive marked jobs J_i and J_{i+1}. Note that s_0 represents the jobs which appear between the beginning of the sequence and job J_1, and $s_{m_j} = \emptyset$. Also, let $S_i = J_i \cup s_i$, $i = 0, 1, 2, \ldots, m_j$ and observe that $S_0 = s_0$. Fig. 1 shows an example sequence where the final job shown is job j, with marked jobs labeled.

As before, B_{J_i}, $i = 0, 1, 2, \ldots, m_j$, represents the starting time in the upper bound sequence of set S_i or equivalently, job J_i.

We use this approach in order to get as much information as possible about the release dates of the jobs, and thus about the lower bound, from the upper bound. Obviously, the jobs in s_i must have release dates greater than B_{J_i}. In addition, it seems intuitively reasonable that most of the jobs in s_i will appear *in the lower bound* by time $B_{J_{i+1}}$. Indeed, one of the objectives of several of the remaining properties in this paper is to characterize the jobs which may be available for processing in the lower bound at time $B_{J_{i+1}}$.

To simplify the proofs, we define the following concepts. For any i, $i = 0, 1, 2, \ldots, m_j - 1$, *at any time* t during the interval from B_{J_i} to $B_{J_{i+1}}$, we define two sets of jobs, set S_t^i and set O_t^i. Set S_t^i contains all of the *released jobs with remaining processing time in the lower bound less than* $p_{J_{i+1}}$ *and greater than* 0, and set O_t^i contains all of the other released jobs with remaining processing time. That is, O_t^i contains released jobs with remaining processing time in the lower bound greater than or equal to $p_{J_{i+1}}$.

Three simple yet crucial observations follow:

1. As jobs are processed in the lower bound, some of them will be moved from set O_t^i to set S_t^i. In particular, a job becomes a *converted job* at time t' if at time t' the job enters $S_{t'}^i$, while it was in O_t^i for all $t < t'$ since its release.
2. At any time h that a job in O_h^i is processing in the lower bound, the set S_h^i must be empty.
3. Given i, $i = 0, 1, 2, \ldots, m_j - 1$ and $t = B_{J_{i+1}}$, $O_t^i \subseteq O_t^{i+1}$, and $S_t^{i+1} \subseteq S_t^i$, since jobs may be short enough to be in S_t^i but too long to be in S_t^{i+1}.

Finally, for any i, $i = 0, 1, 2, \ldots, m_j - 1$, at any time t during the interval from B_{J_i} to $B_{J_{i+1}}$, we define the set $\underline{S}_t^i$ to contain all of the released jobs with remaining processing time in the lower bound less than $p_{J_{i+1}}$ and greater than 0 *except for the jobs released at time t or added to set* S_t^i *at time t*. Thus, the difference between $\underline{S}_t^i$ and S_t^i is that S_t^i may contain jobs added at time t, while $\underline{S}_t^i$ does not. As before, it is easy to see that given i, $i = 0, 1, 2, \ldots, m_j - 1$ and $t = B_{J_{i+1}}$, $\underline{S}_t^{i+1} \subseteq \underline{S}_t^i$.

These definitions are used in the following property, in which we start to characterize the jobs which can be available for processing in the lower bound at any time B_{J_i}.

Property 2.2. *For any job j, and its associated jobs* J_i, $i = 1, 2, \ldots, m_j$ *as defined above, we have that*:

1. *At time* $t = B_{J_i}$, $|\underline{S}_t^i| \leqslant |\underline{S}_t^{i-1}| \leqslant 1$. *That is, the number of jobs with remaining processing time less than* $p_{J_{i+1}}$ *is less than or equal to the number of jobs with remaining processing time less than* p_{J_i} *which is less than or equal to one, not counting jobs which enter the sets at time t.*

2. *The number of jobs which complete in the lower bound during the interval between $B_{J_{i-1}}$ and B_{J_i} is greater than or equal to $|s_{i-1}|$. Furthermore, these jobs are all the jobs in s_{i-1} except for at most one job.*

Proof. Given a job j and its associated jobs J_i, $i=1,2,\ldots,m_j$, the proof of the first item proceeds by induction on i. Clearly, at time B_{J_1}, all of the jobs which appear in the upper bound have also appeared in their entirety in the lower bound. This is true because all of the other jobs released by time B_{J_1} but not started in the upper bound by time B_{J_1} have processing time greater than or equal to p_{J_1} (otherwise they would start at B_{J_1}). This implies that for $t=B_{J_1}$, $|\underline{S}_t^0|=0$ and hence $|\underline{S}_t^1|=0$.

Now, assume that for $i=k$ at time $t=B_{J_k}$, $|\underline{S}_t^k| \leqslant |\underline{S}_t^{k-1}| \leqslant 1$, and the number of jobs which complete in the lower bound during the interval between $B_{J_{i-1}}$ and B_{J_i} is at least $|s_{i-1}|$. Thus, we need to show that at time $t=B_{J_{k+1}}$, $|\underline{S}_t^k|$ is either 0 or 1, which implies that $|\underline{S}_t^{k+1}| \leqslant |\underline{S}_t^k| \leqslant 1$. If, at time $t=B_{J_{k+1}}$, $|\underline{S}_t^k| \leqslant 1$, the proof is complete. If not, find the latest time $\tilde{t}$, $B_{J_k} \leqslant \tilde{t} < B_{J_{k+1}}$ such that $|\underline{S}_{\tilde{t}}^k|=0$ but $|S_{\tilde{t}}^k|>0$.

We first argue that the time $\tilde{t} \geqslant B_{J_k}$ must exist if $|\underline{S}_{B_{J_{k+1}}}^k| > 1$. By contradiction, assume that such a time $\tilde{t} \geqslant B_{J_k}$ does not exist. We consider two cases. If $|\underline{S}_t^k| = |S_t^k| = 0$ for some t, $B_{J_k} \leqslant t < B_{J_{k+1}}$, then either $|\underline{S}_{B_{J_{k+1}}}^k| = 0$, or there exists a time u, $t < u \leqslant B_{J_{k+1}}$ such that $|\underline{S}_u^k|=0$ but $|S_u^k|>0$. Next, consider the case in which $|\underline{S}_t^k|>0$ and $|S_t^k|>0$ for all $B_{J_k} \leqslant t \leqslant B_{J_{k+1}}$. By the induction hypothesis, we know that $|\underline{S}_{B_{J_k}}^k| \leqslant |\underline{S}_{B_{J_k}}^{k-1}| \leqslant 1$. Since *none* of the sets $S_t^k, B_{J_k} \leqslant t \leqslant nB_{J_{k+1}}$ will contain newly added converted jobs (due to the fact that $|S_t^k|>0$ for $B_{J_k} \leqslant t \leqslant B_{J_{k+1}}$) we know that this implies that at most one converted job will be processed during the interval between B_{J_k} to $B_{J_{k+1}}$, and this job must be in set $S_{B_{J_k}}^k$. Of course, this job must have a remaining processing time in the interval between B_{J_k} to $B_{J_{k+1}}$ less than p_{J_k}. This implies that during that interval we have enough time to process this job in the lower bound, plus all the jobs in s_k. Thus, $|\underline{S}_{B_{J_{k+1}}}^k| = 0$, which is a contradiction to our initial assumption. Hence, $\tilde{t}$ exists.

We proceed by considering two cases:

Case 1: If $\tilde{t} \geqslant B_{J_k}$, and there are no converted jobs in set $S_{\tilde{t}}^k$, we note that *none* of the sets S_t^k, $\tilde{t} \leqslant t \leqslant B_{J_{k+1}}$ will contain converted jobs, since at least one of these sets must be empty for a job in O_t^k, $\tilde{t} \leqslant t \leqslant B_{J_{k+1}}$ to process, but after time $\tilde{t}$ this doesn't happen. Furthermore, we know that all of the jobs in s_k which were released before time $\tilde{t}$ have completed processing in the lower bound, since at time $\tilde{t}$, $|\underline{S}_{\tilde{t}}^k|=0$. Thus, only jobs in s_k released at or after time $\tilde{t}$ are in the sets $S_t^k, \tilde{t} \leqslant t \leqslant B_{J_{k+1}}$. In addition, we know that all of these jobs completed processing in the lower bound, since they completed processing in the upper bound, and thus $|\underline{S}_{B_{J_{k+1}}}^k| = 0$.

Case 2: If $\tilde{t} \geqslant B_{J_k}$, and there is a converted job in set $S_{\tilde{t}}^k$ (clearly, there cannot be more than one because the lower bound will always start processing the first converted job), we note that *none* of the sets $S_t^k, \tilde{t} \leqslant t \leqslant B_{J_{k+1}}$ will contain any additional converted jobs, since at least one of these sets must be empty for a job in O_t^k, $\tilde{t} \leqslant t \leqslant B_{J_{k+1}}$ to process, but after time $\tilde{t}$ this doesn't happen. Since, as we stated above, only jobs in s_k released at or after time $\tilde{t}$ are in the sets S_t^k, $\tilde{t} \leqslant t \leqslant B_{J_{k+1}}$ and since all of the jobs in set s_k released at or after time $\tilde{t}$ had enough time to process in the lower bound since they completed processing in the upper bound, we know that either all of them, or all of them except one plus the one converted job, complete by time $B_{J_{k+1}}$, and hence $|\underline{S}_{B_{J_{k+1}}}^k| \leqslant 1$.

The second part of the property follows from the proof above, since either all of the jobs in s_k complete processing in the lower bound, or all the jobs in s_k but one complete processing, and instead a converted job completes processing in the lower bound. □

The following properties are also used throughout the remainder of the proof.

Property 2.3. *Consider an SPTA schedule, and an SPTR schedule, generated as described above. At any time t, the number of jobs which have completed processing in the SPTR schedule is greater than or equal to the number of jobs which have completed in the SPTA schedule.*

The proof of the property follows by induction, utilizing the definition of the SPTR strategy.

Property 2.4. *For any job j, and its associated jobs* $J_i,\ i=1,2,\ldots,m_j$ *as defined above, if at time* $t=B_{J_i}$ *either job* J_i *has not yet been completely processed in the lower bound or* $|\underline{S}^i_t|=1$, *at least* $|s_i|+1$ *jobs will complete in the lower bound in the interval between* B_{J_i} *and* $B_{J_{i+1}}$.

Proof. First, consider the case in which only the jobs in s_i and either job J_i or the job in $\underline{S}^i_t$, $t=B_{J_i}$, are available during this interval. In this case, at least $|s_i|+1$ jobs will complete in this interval, since all of the jobs in $|s_i|$ and the one additional job with remaining processing time no more than p_{J_i} (and perhaps others) are available to fill the time between B_{J_i} and $B_{J_{i+1}}$. Now for the case in which additional jobs are available, by the SPTR strategy, these additional jobs will only be processed if they complete before the jobs they interrupt. Thus, with these additional jobs available, either the same number or more jobs will complete in the lower bound. □

Property 2.5. *For any job j, and its associated jobs* $J_i,\ i=1,2,\ldots,m_j$ *as defined above, if at time* $t=B_{J_i}$, $|\underline{S}^i_t|=1$, *the job in* $\underline{S}^i_t$ *will complete processing in the lower bound by time* $t=B_{J_{i+1}}$.

Proof. This property follows from Property 2.4, and the fact that all jobs released between time $t=B_{J_i}$ and time $t=B_{J_{i+1}}$ are in set s_i, or have processing times greater than or equal to $p_{J_{i+1}}$. □

For the next property, we need the following definitions. By Property 2.3, between time B_{J_1} and time $B_{J_k},\ k=1,2,\ldots,m_j$, in the lower bound at least $\sum_{i=1}^{k-1}|s_i|+(k-1)$ jobs must have completed in the lower bound, since that number *exactly* have completed in the upper bound. Order the jobs which have completed in the lower bound by time B_{J_k} from smallest to largest (breaking ties arbitrarily), and let set L_k contain the $\sum_{i=1}^{k-1}|s_i|+(k-1)$ smallest jobs which have completed in the lower bound between time B_{J_1} and time B_{J_k}. Also, let set $U_k=(\bigcup_{i=1}^{k-1}s_i)\cup(\bigcup_{i=2}^{k}J_i)$, the set of jobs which completes in the upper bound between the completion of job J_1 and the completion of job J_k.

Note that any job which completes in the lower bound by time B_{J_k} and is not included in set L_k must have a processing time greater than or equal to p_{J_k}. This is true since by the way set L_k is created, all of the potentially shorter jobs are in the set.

Using the above definitions, we show that:

Property 2.6. *For any job j, its associated jobs* $J_i,\ i=1,2,\ldots,m_j-1$, *and their associated sets* U_i *and* L_i *as defined above, it is possible to create a one to one matching of each job a in* U_i *with a unique job b in* L_i *such that* $p_b \geqslant p_a$ *for each of the pairs.*

This property has an important implication: Since L_k contains the $\sum_{i=1}^{k-1}|s_i|+(k-1)$ smallest jobs which have completed in the lower bound between time B_{J_1} and time B_{J_k}, the sum of the processing times of the jobs in this set is a lower bound on the completion time of the first $\sum_{i=1}^{k-1}|s_i|+(k-1)$ which complete in this interval. Furthermore, by matching these jobs with shorter jobs in the upper bound, we know that the processing time for the first $\sum_{i=1}^{k-1}|s_i|+(k-1)$ which complete in the lower bound during the interval between B_{J_1} and B_{J_k} is longer than the processing time in the upper bound between the completion of job J_1 and the completion of job J_k.

Proof. Given a job j and its associated jobs $J_i,\ i=1,2,\ldots,m_j$, the proof proceeds by induction on i. First, note that all of the jobs in s_0, and no others, appear completely in the lower bound by time B_{J_1}. We know that at least $|s_1|+1$ jobs complete in the lower bound in the interval between B_{J_1} and B_{J_2}, by Property 2.3. Furthermore, the second part of Property 2.2 tells us that all but one of the jobs in s_1 complete during this interval. Thus, the jobs which complete in the lower bound in the interval from B_{J_1} to B_{J_2} must be *either* all of the jobs in set s_1 plus *at least* one job with processing time in the interval $[p_{J_1}, p_{J_2}]$, *or* all of the jobs in set s_1 except one plus *at least* two jobs with processing time in the interval $[p_{J_1}, p_{J_2}]$ (because shorter jobs would be in s_1, and longer jobs will not process when shorter jobs are available). In either case, clearly, the property holds at time B_{J_2}.

Now, assume the property holds for $B_{J_i},\ i=1,2,\ldots,k$, and consider $B_{J_{k+1}}$. We consider three cases:

Case 1: Job J_k has not completed processing in the lower bound by time B_{J_k}, and $|\underline{S}^k_{B_{J_k}}|=0$. From Property 2.4, we know that at least $|s_k|+1$ must complete

in the lower bound in the interval from B_{J_k} to $B_{J_{k+1}}$. These jobs must be *either* all of the jobs in set s_k plus *at least* one job with total processing time greater than or equal to $p_{J_{k+1}}$, *or* all of the jobs in set s_k except one plus *at least* two jobs with processing time greater than or equal to $p_{J_{k+1}}$. This must be true because shorter jobs would be in s_k. In either case, clearly, the property holds for $B_{J_{k+1}}$, since the jobs available for selection to L_{k+1} are all of the jobs in L_k, either some or all of the jobs in s_k, and some additional jobs with processing times greater than $p_{J_{k+1}}$.

Case 2: Job J_k has completed processing in the lower bound by time B_{J_k}, and $|\underline{S}^k_{B_{J_k}}| = 0$. If $|s_k| + 1$ or more jobs complete in the lower bound in the interval from B_{J_k} to $B_{J_{k+1}}$, these jobs must be *either* all of the jobs in set s_k plus *at least* one job with total processing time greater than or equal to $p_{J_{k+1}}$, *or* all of the jobs in set s_k except one plus *at least* two jobs with processing time greater than or equal to $p_{J_{k+1}}$ for the same reason as in *Case* 1. Thus, the proof follows as in *Case* 1. If only $|s_k|$ jobs complete in the lower bound in the interval from B_{J_k} to $B_{J_{k+1}}$, we know from Property 2.3 that at time B_{J_k} at least one job not in L_k must have completed in the lower bound. Otherwise, at time $B_{J_{k+1}}$, it would be impossible for Property 2.3 to hold. Furthermore, we have already observed that jobs which complete in the lower bound by time B_{J_k} and are not in L_k must have processing time of at least p_{J_k}. Thus, the jobs added to U_k to get U_{k+1} are $s_k \cup J_{k+1}$, while the jobs available for the selection of L_{k+1} are all of the jobs in L_k, at least one additional job with processing time of at least p_{J_k}, and *either* all of the jobs in s_k *or* all of the jobs in s_k except one plus one job with processing time greater than or equal to $p_{J_{k+1}}$. In either case, clearly the property holds for $B_{J_{k+1}}$.

Case 3: $|\underline{S}^k_{B_{J_k}}| = 1$. We call the job in $\underline{S}^k_{B_{J_k}}$ job x. By Property 2.4, we know that at least $|s_k| + 1$ jobs complete in the lower bound in the interval from B_{J_k} to $B_{J_{k+1}}$, and by Property 2.5, we know that job x completes in the lower bound in this interval. Consider the following two cases:

- If job x is not in $\bigcup_{i=1}^{k-1} s_i$, it has total processing time of at least p_{J_k}.
- If job x is in $\bigcup_{i=1}^{k-1} s_i$, we know that in U_k it was matched with a job in L_k with processing time of at least p_{J_k}.

Thus, in both cases, the proof follows as in Case 1. □

Now, for any job j, and its associated jobs J_i, $i = 2, 3, \ldots, m_j$ as defined above, number the jobs in the lower bound *in the order of their completion*, and let F_k be the time that the $(|s_0| + \sum_{i=1}^{k-1} |s_i| + (k-1))$th job completes in the lower bound. Note that, this is not necessarily the time that the $(|s_0| + \sum_{i=1}^{k-1} |s_i| + (k-1))$th job completes in the upper bound, which is by definition B_{J_k}. Also, recall that the $(|s_0| + \sum_{i=1}^{k-1} |s_i| + (k-1))$th job to complete in the upper bound is not necessarily the same job as the $(|s_0| + \sum_{i=1}^{k-1} |s_i| + (k-1))$th to complete in the lower bound. We show that:

Property 2.7. *For any job j, and its associated jobs J_i, $i = 1, 2, \ldots, m_j - 1$ as defined above, we have that $B_{J_i} - F_i \leqslant p_{J_1} - p_{J_i}$.*

Proof. First, let $p(U_k)$ and $p(L_k)$ equal the sum of the processing times of the jobs in U_k and L_k, respectively. Recall that $U_k = (\bigcup_{i=1}^{k-1} s_i) \cup (\bigcup_{i=2}^{k} J_i)$, so that

$$B_{J_k} = p(U_k) + p_{J_1} - p_{J_k}$$

also, clearly $p(L_k) \leqslant F_k$, and $p(U_k) \leqslant p(L_k)$.

Hence,

$$\begin{aligned} B_{J_k} - F_k &\leqslant p(U_k) + p_{J_1} - p_{J_k} - p(L_k) \\ &\leqslant p(U_k) + p_{J_1} - p_{J_k} - p(U_k) \\ &= p_{J_1} - p_{J_k}. \qquad \square \end{aligned}$$

Using these properties, we complete the proof of Theorem 1.1 as follows. From Property 2.1, it is clear that we only have to prove the theorem for a single chain, and it will generalize to the situation for many chains. We number the jobs in the chain consecutively from 1 to n'. Clearly, the theorem holds for jobs 1 and n'. For each of the remaining jobs i, $i = 2, 3, \ldots, n' - 1$, apply Property 2.7 with $j = i + 1$. □

3. Proof of the corollary

Applying Theorem 1.1, we get that

$$Z^{\mathrm{SPTR}} \leqslant Z^* \leqslant Z^{\mathrm{SPTA}} \leqslant Z^{\mathrm{SPTR}} + n\bar{P}.$$

Noting that $\bar{P}$ has a finite bound, dividing by n^2, and taking n to infinity completes the proof of the corollary. □

4. Future extensions

Our motivation for analyzing the single machine model with release dates stems from our interest in general flow shop problems. Indeed, in [10,11], we utilize the fact that the single machine mean completion time problem *without release dates* is trivially solved using SPT ordering to identify asymptotically optimal algorithms for the flow shop mean completion time problem *without release dates*. Specifically, in [10,11] we prove that the SPT ordering is asymptotically optimal if the job processing times are independently and identically distributed across jobs and across machines.

The challenge, of course, is to extend these results to a more general setting *with release dates*. This implies that, we need to relate the flow shop mean completion time problem with release dates to the single machine mean completion time problem with release dates. The current paper, thus, plays an important role in this line of research. It identifies a simple, asymptotically optimal, on-line algorithm for the single machine problem. In a companion paper, we will report our finding on the general flow shop mean completion time with release dates.

Acknowledgements

We are thankful for the anonymous referee's thorough review and valuable suggestions.

References

[1] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, M. Sviridenko, Approximation schemes for minimizing average weighted completion time with release dates, Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 1999, pp. 32–43.

[2] H. Beloudadah, M.E. Posner, C.N. Potts, Scheduling with release dates on a single machine to total weighted completion time, Discrete Appl. Math. 36 (1992) 213–231.

[3] L. Bianco, S. Ricciardelli, Scheduling of a single machine to minimize total weighted completion time subject to release dates, Nav. Res. Logistics Quart. 29 (1982) 151–167.

[4] C. Chekuri, R. Motwani, B. Natarajan, C. Stein, Approximation techniques for average completion time scheduling, Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 1997, pp. 609–618.

[5] M.I. Dessouky, J.S. Deogun, Sequencing jobs with unequal ready times to minimize mean flow time, SIAM J. Comput. 10 (1981) 192–202.

[6] M.E. Dyer, L.A. Wolsey, Formulating the single machine sequencing problem with release dates as a mixed integer program, Discrete Appl. Math. 26 (1990) 255–270.

[7] Gazmuri, Probabilistic analysis of a machine scheduling problem, Math. Oper. Res. 10 (1985) 328–339.

[8] A.M.A. Hariri, C.N. Potts, An algorithm for single machine sequencing with release dates to minimize total weighted completion time, Discrete Appl. Math. 5 (1983) 99–109.

[9] J.A. Hoogeveen, A.P.A. Vestjens, Optimal on-line algorithms for single-machine scheduling, Proceedings of the Fifth Conference on Integer Programming and Combinatorial Optimization, 1996, pp. 404–414.

[10] P. Kaminsky, D. Simchi-Levi, Probabilistic analysis and practical algorithms for the flow shop weighted completion time problem, Oper. Res. 46 (1997) 872–882.

[11] P. Kaminsky, D. Simchi-Levi, The asymptotic optimality of the SPT rule for the flow shop mean completion time problem, Oper. Res. 49 (2001) 293–304.

[12] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, Ann. Discrete Math. 1 (1977) 343–362.

[13] C. Phillips, C. Stein, J. Wein, Scheduling jobs that arrive over time, Math. Programming B 82 (1998) 199–223.

[14] M. Pinedo, Scheduling: Theory, Algorithms and Systems, Prentice-Hall, Englewood Cliffs, NJ, 1995.

[15] L. Schrage, A proof of the optimality of the shortest remaining processing time discipline, Oper. Res. 16 (1968) 687–690.

[16] J.P. Sousa, L.A. Wolsey, A time indexed formulation of non-preemptive single machine scheduling problems, Math. Programming 54 (1992) 353–367.

[17] J.M. van den Akker, LP-based solution methods for single-machine scheduling problems, Ph. D. Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1994.