# Bicriteria Job Sequencing with Release Dates

Yaoguang Wang
Max-Planck-Institut für Informatik
Im Stadtwald, D-66123 Saarbrücken, Germany
Email: wang@mpi-sb.mpg.de

November 2, 1996

**Abstract**

We consider the single machine job sequencing problem with release dates. The main purpose of this paper is to investigate efficient and effective approximation algorithms with a bicriteria performance guarantee. That is, for some $(\rho_1, \rho_2)$, they find schedules simultaneously within a factor of $\rho_1$ of the minimum total weighted completion times and within a factor of $\rho_2$ of the minimum makespan.

The main results of the paper are summarized as follows. First, we present a new $O(n \log n)$ algorithm with the performance guarantee $\left(1 + \frac{1}{\beta}, 1 + \beta\right)$ for any $\beta \in [0, 1]$. For the problem with integer processing times and release dates, the algorithm has the bicriteria performance guarantee $\left(2 - \frac{1}{p_{max}}, 2 - \frac{1}{p_{max}}\right)$, where $p_{max}$ is the maximum processing time. Next, we study an elegant approximation algorithm introduced recently by Goemans. We show that its randomized version has expected bicriteria performance guarantee $(1.7735, 1.51)$ and the derandomized version has the guarantee $(1.7735, 2 - \frac{1}{p_{max}})$.

To establish the performance guarantee, we also use two LP relaxations and some randomization techniques as Goemans does, but take a different approach in the analysis, based on a decomposition theorem. Finally, we present a family of bad instances showing that it is impossible to achieve $\rho_1 \leq 1.5$ with this LP lower bound.

## 1 Introduction

We are given a set $N$ of $n$ jobs to be processed on a single machine. Each job $j \in N$ has a required processing time $p_j$, a nonnegative weight $w_j$, and a release date $r_j$ meaning that it cannot start processing before $r_j$. The problem of finding a processing sequence to minimize the total weighted completion times $\sum_{j \in N} w_j C_j$ is known to be strongly NP-hard. The existing exact optimization algorithms can typically handle problems of size less than 50 jobs. In recent years, there has been exciting progress in finding good $\rho$-approximation algorithms with performance guarantee $\rho$, that is, the algorithm can find a schedule in polynomial time with the objective value no worse than $\rho$ times the optimal value. We refer readers to [1, 4] for an overview on this subject.

More recently, Goemans (1996) introduced an elegant algorithm and showed that it achieves the performance guarantee of 2, which significantly improves upon the previous best known 3-approximation algorithms in both on-line and off-line setting. His approach exploits some nice properties of *optimal solutions* to *two LP relaxations*. The first one is is a time-indexed formulation due to Dyer and Wolsey[2] (DW-formulation for short):

$$
\begin{array}{llll}
(1) & wC^* \doteq & \min \sum_{j=1}^{n} w_j C_j & \\
& \text{s.t.} & \sum_{j=1}^{n} y_{jt} \le 1, & t = 1, \ldots, \hat{t}, \\
& & \sum_{t=r_j}^{\hat{t}} y_{jt} = p_j, & j = 1, \ldots, n, \\
& & C_j = \frac{p_j}{2} + \frac{1}{p_j} \sum_{t=r_j}^{\hat{t}} (t + \frac{1}{2}) y_{jt}, & j = 1, \ldots, n, \\
& & 0 \le y_{jt} \le 1, & j = 1, \ldots, n,\ t = r_j, \ldots, \hat{t},
\end{array}
$$

where $\hat{t}$ is an upper bound on the makespan of the problem; the second one is

$$
\begin{array}{lll}
(2) \quad wC^* & \doteq & \min \sum_{j \in N} w_j C_j \\
& \text{s.t.} & \sum_{j \in S} p_j C_j \ge r(S)p(S) + g(S), \ \ \forall S \subseteq N,
\end{array}
$$

where $r(S) = \min\{r_j : j \in S\}$, $p(S) = \sum_{j \in S} p_j$, $g(S) = \frac{1}{2}[(\sum_{j \in S} p_j)^2 + \sum_{j \in S} p_j^2]$. It forms a supermodular relaxation as shown in his early work [4].

In this paper, we investigate efficient and effective approximation algorithms with a bicriteria performance guarantee. That is, for a pair of small scalars $(\rho_1, \rho_2)$, the algorithm always finds a schedule simultaneously within a factor of $\rho_1$ of the minimum total weighted completion time and within a factor of $\rho_2$ of the minimum makespan. We remark that a by-product of the Goemans's $A_\alpha$ algorithm has the bicriteria guarantee of $\left(\max\left\{1 + \frac{1}{\alpha}, 2\alpha + 1\right\}, 1 + \alpha\right)$.

Our first main result is a new $O(n \log n)$ scheduling algorithm with the bicriteria guarantee $\left(1 + \frac{1}{\beta}, 1 + \beta\right)$, where $\beta$ is any input parameter with $\beta \in [0, 1]$. For the problem with integer processing times and release dates, the algorithm has the bicriteria performance guarantee $\left(2 - \frac{1}{p_{max}}, 2 - \frac{1}{p_{max}}\right)$, where $p_{max}$ is the maximum processing time. Second, we derive improved performance guarantees for the Goemans's algorithms. We show that $RANDOM_\alpha$ has the expected bicriteria performance guarantee $(1.7735, 1.51)$ in both on-line and off-line settings, and $Best_\alpha$ has the performance guarantee $(1.7735, 2 - \frac{1}{p_{max}})$. Finally, we provide a family of bad instances showing that it is impossible to derive the performance guarantee $\rho_1 \le 1.5$ for approximation algorithms with this LP lower bound.

To establish the performance guarantee, we also use the two LP relaxation, but take a different approach in the analysis based on a *decomposition theorem* for a slightly altered problem from the original one. Specifically, in the preemptive scheduling with the DW-formulation, we derive a new problem with the release date of each job as its start time in the preemptive schedule. Notice that the optimal objective LP value of the new problem is the same as that of the original one, while every feasible schedule of the new problem is also feasible for the original one with the same objective value. This implies that it suffices to establish the performance guarantees for the new problem as we use the LP value for the lower bound. The new problem is then represented by a *decomposition tree* with nontrivial subtrees corresponding to its subproblems. Consequently, we establish our performance guarantees by induction on the height of nodes representing the subproblems. This approach also works for deriving improved approximation ratios for the Goemans's algorithms with the help of the following probability density function $\xi(x)$ over $x \in [0,1]$:

$$(3) \quad \xi(x) = \begin{cases} 0.7179(1+x)e^{0.34067x}, & \text{if } 0 \le x < 0.454628; \\ \frac{1.7735}{1+x}, & \text{if } 0.454628 \le x \le 1. \end{cases}$$

In the following sections, we present our main results and a brief description of the algorithm. We shall stress on main ideas and techniques used to achieve these results. We provide the detailed algorithms with examples in Appendix A, and long, technical proofs in Appendix B.

## 2 A decomposition theorem

The following notation and definitions are used in the paper. Given an optimal LP solution $C$, a subset $S \subseteq N$ is said to be *tight* w.r.t. $C$ if $\sum_{j \in S} p_j C_j = r(S)p(S) + g(S)$. If $S = \{j\}$ is a singleton tight set, i.e., $p_j C_j = r_j p_j + p_j^2$, it is called a *trivial tight set.* we use $P(w, p, r, S)$, $S \subseteq N$, denote the problem defined on $S$. Let $w(S) = \sum_{j \in S} w_j$ for $S \subseteq N$.

We first establish some interesting properties on the tight sets and their associated release dates. These properties are crucial to derive a decomposition theorem and establish bicriteria performance guarantees.

**Lemma 2.1 ($r$-lifting lemma)** *Let $S_1$ and $S_2$ be any two tight sets satisfying $S_1 \setminus S_2 \neq \emptyset$ and $S_2 \setminus S_1 \neq \emptyset$. Then*

$$r(S_1) \le r(S_2) \Longrightarrow r(S_1) + p(S_1 \setminus S_2) \le r(S_2).$$

**Lemma 2.2** *Let $S$ be the union of disjoint tight sets $S_1, \dots, S_k$ with $r(S_1) \le \cdots \le r(S_k)$. Then*

$$(4) \quad r(S_{i-1}) + p(S_{i-1}) \le r(S_i), \quad i = 2, \dots, k.$$

*Moreover, $S$ is tight if and only if the equality holds for each $i$ in (4).*

With Lemma 2.1, we can derive the following result.

**Theorem 2.3** *If two tight sets $S_1$ and $S_2$ intersect properly with $r(S_1) \le r(S_2)$, then all sets $S_1 \setminus S_2$, $S_1 \cap S_2$, and $S_1 \cup S_2$ are tight.*

As an immediate consequence, we have

**Corollary 2.4** *There exists a family of nested tight sets that covers $N$.*

Note that the above corollary also follows from the Goemans's result.

By employing Dyer and Wolsey's formulation [2], Goemans [4] suggested a nice $O(n \log n)$ preemptive scheduling algorithm, which constructs an optimal LP solution. For our purposes, we need additional work for (i) the $O(n \log n)$ time implementation of the algorithm and (ii) the analysis of the bicriteria performance guarantee. In particular, the second task requires finding a new vector $r'$ of release dates such that a new problem $P(w, p, r', N)$ induced by $r'$ has the following properties:

1. $r'_j$ is the start time of job $j$ in the preemptive schedule. So $r'_j \geq r_j$ for all $j \in N$,

2. the optimal LP value for $P(w, p, r', N)$ remains the same as the original one.

3. each nontrivial minimal tight set $Q$ with respect to $\sum_{j \in C} p_j C_j = r'(Q)p(Q) + g(Q)$ can be expressed as a partition $Q = (\{q\}, S_1, \ldots, S_k)$, where each $S_i$ is a minimal tight set except the singleton set $\{q\}$ and

   $r'_q = r'(Q) < r'(S_1) \leq \cdots \leq r'(S_k) < r'(Q) + p(Q) - p(S_k)$.

We call such sets $Q$ *blocks*.

We present a preemptive scheduling algorithm, called **D-Tree**, along with a detailed example in the appendix. **D-Tree** extends the Goemans's preemptive scheduling algorithm with an additional task for construction of a decomposition tree $T$ representing $P(w, p, r', N)$. **D-Tree** runs in $O(n \log n)$ time, and it is called only once by the main algorithm $GP(\beta)$. Since the application of **D-Tree** to $P(w, p, r', N)$ yields the same LP solution $C$, $r'$ satisfies properties 1 and 2 above. Observe that any feasible schedule to $P(w, p, r', N)$ is also feasible for $P(w, p, r, N)$, by property 2 it suffices to establish the performance guarantees with respect to $P(w, p, r', N)$.

To see that property 3 also holds, consider any preempted job $q$ with its associated time window $[r'_q, c_q]$, where $c_q$ denotes the time at which job $q$ completes processing. Let $Q$ be the set of all jobs processed within the time window. Consider any job $j \in Q \setminus \{q\}$, and let $t_j$ and $c_j$ denote its respective start time and completion time in the preemptive schedule. By construction and the nestedness property of tight sets, we have $t_j > r'_q$ and $c_j < c_q$. Further, since by construction, the machine must always be busy processing jobs in $Q$ within the time window, by the results in [3], $Q$ is a tight set. It then follows that $Q$ forms a block.

Before presenting the decomposition theorem, we state an easy result. Here we use the Goemans's notion of separable sets. A set $S \subseteq N$ is said to be *separable* if $S$ can be expressed as a partition $S = (S_1, \ldots, S_k)$ of tight sets $S_i$'s.

**Lemma 2.5** *Suppose that $N$ is separable. Let $\rho_2 = 1 + \beta$ and $\rho_1 \geq 1 + \beta$ with $\beta \in [0, 1]$. If an algorithm $A(\rho_1, \rho_2)$ can find subschedules for all subproblems of $P(w, p, r', N)$ with the bicriteria performance guarantee $(\rho_1, \rho_2)$, then it can also construct a schedule for $P(w, p, r', N)$ from these subschedules with the same performance guarantee.*

For the cases of $N$ inseparable, we have the following decomposition theorem.

**Theorem 2.6 (Decomposition theorem)** *Suppose that $N$ is inseparable. Let $C$ be an optimal LP solution obtained from the preemptive scheduling algorithm. Then, there exists a nested family $\mathcal{F}$ of $n$ minimal tight sets $S$ with respect to $\sum_{j \in S} p_j C_j = r(S)p(S) + g(S)$, $S \subseteq N$, such that*

- *the unpreempted jobs are precisely those singleton sets in $\mathcal{F}$;*

- *there exists a one-to-one correspondence between the preempted jobs $q$ and blocks $Q \in \mathcal{F}$ of the form $Q = (\{q\}, S_1, \ldots, S_k)$.*

*Furthermore, $\mathcal{F}$ can be represented by a decomposition tree $T$ with each nontrivial subtree (neither $T$ nor a leaf of $T$) corresponding to a block.*

An example, together with the preemptive schedule and its associated decomposition tree, is given in **Appendix A.** Note that each subproblem $P(w, p, r', Q)$ with $Q = (\{q\}, S_1, \ldots, S_k)$ is uniquely *represented by node* $q$ in $T$. Note also that $\mathcal{F}$ defines an optimal dual basis.

**Corollary 2.7** *Any minimal nontrivial tight set $Q$ induces a block.*

We need more notation and expressions for our analysis. For any block $Q = (\{q\}, S_1, \ldots, S_k)$, define

$$
\begin{aligned}
(5) \quad \alpha_0 = 0, \quad a_1 &= \alpha_1 = \frac{r'(S_1) - r'_q}{p_q}, \qquad a_{k+1} = \frac{r'_q + p(Q) - r'(S_k) - p(S_k)}{p_q}, \\
a_j &= \frac{r'(S_j) - r'(S_{j-1}) - p(S_{j-1})}{p_q}, \quad \alpha_j = \sum_{i=1}^{j} a_i, \quad j = 2, \ldots, k+1.
\end{aligned}
$$

We may interpret $a_j$ as the fraction of job $q$ performed between $S_{j-1}$ and $S_j$, and $\alpha_j$ as the fraction of job $q$ completed before the start of $S_j$. By definition $\alpha_{k+1} = 1$.

By construction of the preemptive schedule, we may write $r'(S_i)$ and $C_q$ as (A proof for (7) is given in Appendix B):

$$
(6) \quad r'(S_h) = r'(Q) + p_q \sum_{j=1}^{h} a_j + \sum_{j=1}^{h-1} p(S_j) = r'(Q) + p_q \alpha_h + \sum_{j=1}^{h-1} p(S_j), \ h = 1, \ldots, k;
$$

$$
(7) \quad C_q = r'(Q) + p_q + \sum_{j=2}^{k+1} a_j \sum_{i=1}^{j-1} p(S_j) = r'(Q) + p_q + \sum_{j=1}^{k} (1 - \alpha_j) p(S_j).
$$

These two expressions will be used later in our analysis.

## 3 A bicriteria scheduling algorithm $GP(\beta)$

We briefly describe how our algorithm $GP(\beta)$ constructs the required schedule with $T$. The detailed algorithm with the example is presented in the appendix. The algorithm begins with nodes of *height* 1 in $T$. (The *height* of a node $v$ is the length of a longest path in $T$ from $v$ to a leaf. All leaves have height 0.) When the algorithm begins working on an internal node $q$ of height $i$ with its associated block $Q = (\{q\}, S_1, \ldots, S_k)$, all subproblems represented by nodes of height less than $i$ are solved. Thus, the processing order for jobs inside each $S_i$ is available, and the processing sequence for $S_i$'s is set as $S_1 \to \cdots \to S_k$ (i.e., no jobs in $S_j$ can start before all jobs in $S_{j-1}$ are completed.) The main steps of $GP(\beta)$ include: (i) updating the block information for later use by its unique predecessor in $T$; (ii) computing the LP completion time $C_q$ for job $q$, and (iii) inserting job $q$ at $h$ (i.e., sequencing $q$ as $S_{h-1} \to q \to S_h$) with the minimum $B(h, \beta)$ (a position function defined below.) Repeat until the processing sequence for all jobs is determined.

Let $h(\beta) = \min\{h : \alpha_h > \beta\}$. For $h = 1, \ldots, h(\beta)$, define the following *position function*:

$$(8) \quad B(h,\beta) = p_q \left[ \sum_{j=1}^{h-1} a_j \sum_{i=j}^{h(\beta)} w(S_i) + \sum_{i=h}^{h(\beta)} w(S_i) \right] + w_q \left[ p_q \left( 1 + \sum_{j=1}^{h-1} a_j \right) + (1+\beta) \sum_{j=1}^{h-1} p(S_j) \right].$$

Now, we state the main results of this section.

**Theorem 3.1** *For any $\beta \in [0,1]$, the algorithm $GP(\beta)$ delivers a schedule with the bicriteria performance guarantee $\left(1 + \frac{1}{\beta}, 1+\beta\right)$ in $O(n \log n)$ time.*

**Theorem 3.2** *Assume that all processing times and release dates are integers, and $p_{max} = \max\{p_j : j \in N\}$. Let $\beta = 1 - \frac{1}{p_{max}}$. Then $GP(\beta)$ constructs a schedule $\hat{C}$ with the bicriteria guarantee $\left(2 - \frac{1}{p_{max}}, 2 - \frac{1}{p_{max}}\right)$.*

The bicriteria performance guarantee can be established by induction. Suppose that the bicriteria performance guarantee is valid for all subproblems represented by $q$ of height less than $i$ in $T$. We now consider any problem represented by a node $q$ of height $i$ with its associated block $Q = (\{q\}, S_1, \ldots, S_k)$. Recall by Lemma 2.2 that the sequencing order of the tight sets can be set as $S_1 \to \cdots \to S_k$, and we need to find a position $h$ for $q$. For this purpose, we define $H^h$ as a schedule of sequencing $q$ (at $h$) as $\cdots S_{h-1} \to q \to S_h \cdots$, for $h = 1, \ldots, k+1$. We use the following two propositions to establish the guarantee $\rho_2$. For $\rho_1$, we use a random choice of $h$ and establish $E[wH^h] \le (1+\beta)\sum_{j\in Q} w_j C_j$. The bicriteria performance guarantee for $P(w,p,r',N)$ then follows by induction.

Without loss of generality, we may assume that $r'_q = r'(Q) = 0$. Let $\beta^* = \alpha_{h(\beta)}$, and note by definition of $h(\beta)$, $\beta^* \ge \beta$. In the proofs of the above two theorems, we use the following intermediate results.

**Lemma 3.3** *Suppose that we sequence $q$ at $h$ with probability $\frac{a_h}{\beta^*}$. Then*

$$E[B(h,\beta)] \le \left(1 + \frac{1}{\beta^*}\right) p_q \sum_{j=1}^{k} a_j \sum_{i=1}^{k} w(S_i) + w_q \left[ p_q \left(1 + \frac{\beta^*}{2}\right) + \left(1 + \frac{1}{\beta^*}\right) \sum_{j=2}^{k+1} a_j \sum_{i=1}^{j-1} p(S_i) \right].$$

**Proposition 3.4** *For any $h \le h(\beta)$, the makespan of the schedule $H^h$ for $P(w, r', Q)$ is bounded by $(1+\beta)p(Q)$.*

**Proposition 3.5** *Assume that all processing times and release dates are integers. Let $\beta = 1 - \frac{1}{p_{max}}$. Then the following statements are true.*

1. *$h(\beta) = k+1$ and $\beta^* = 1$;*

2. *For any $h \le h(\beta)$, the makespan of the schedule $H^h$ is at most $\left(2 - \frac{1}{p_{max}}\right) p(Q)$.*

## 4 Bicriteria performance guarantees for the Goemans's algorithms

Goemans recently introduced an elegant algorithm $A_\alpha$ for constructing a schedule based on the preemptive schedule. For any $\alpha \in [0,1]$, let $t_j(\alpha)$ be the time at which $\alpha p_j$ units of job $j$ has been processed. The $A_\alpha$ algorithm is to schedule jobs as early as possible in the order of $t_j(\alpha)$.

Let $C^\alpha$ be the schedule obtained from the $A_\alpha$ algorithm. For any tight set $S$, let $r^\alpha(S) = \max\{C_j^\alpha : j \to S\}$ denote the completion time of the last job before the start of the first job in $S$. Define $r^\alpha(N) = 0$.

A useful observation implicitly used in the analysis is that any job $j \in N \setminus S$ is sequenced as either $j \to S$ or $S \to j$ in $C^\alpha$ for any $\alpha \in [0,1]$. Furthermore, from Proposition 7 in [4], we derive that for any tight set $S$, the jobs in $S$ can be scheduled between $r^\alpha(S)$ and $r^\alpha(S) + (1+\alpha)p(S)$.

Now, consider any block $Q = (\{q\}, S_1, \ldots, S_k)$. For each $h = 1, \ldots, k$, we derive

$$
\begin{aligned}
(9) \quad r^\alpha(S_h) &\le r^\alpha(Q) + (1+\alpha)\sum_{j=1}^{h-1} p(S_j) \\
&\quad + p_q \sum_{j=1}^{h} I(\alpha_{j-1} < \alpha \le \alpha_j)\left(1 + \sum_{i=1}^{j-1} a_i\right) + p_q \alpha_h I(\alpha > \alpha_h) \\
&= r^\alpha(Q) + (1+\alpha)\sum_{j=1}^{h-1} p(S_j) + p_q I(\alpha \le \alpha_h) + \sum_{j=1}^{h} p_q a_j I(\alpha > \alpha_j), \quad \forall \alpha \in [0,1],
\end{aligned}
$$

where $I(.)$ is an indicator function. The completion time of job $q$ is

$$
\begin{aligned}
(10) \quad C_q^\alpha &\le r^\alpha(Q) + \sum_{j=1}^{k+1} I(\alpha_{j-1} < \alpha \le \alpha_j)\left\{p_q + \sum_{i=1}^{j-1}[p_q a_i + (1+\alpha)p(S_i)]\right\} \\
&= r^\alpha(Q) + p_q + \sum_{j=1}^{k} p_q a_j I(\alpha > \alpha_j) + \sum_{j=1}^{k}(1+\alpha)p(S_j) I(\alpha > \alpha_j), \quad \forall \alpha \in [0,1].
\end{aligned}
$$

We briefly describe how to establish the performance guarantee with the help of Theorem 2.6. By Lemma 2.5, we only need to consider $N$ inseparable. Let $U$ and $V$ denote the set of unpreempted and the set of preempted jobs, respectively. Notice that for any $u \in U$, there exists a unique minimal block $Q^u = (\{q^u\}, S_1^u, \ldots, S_k^u)$ containing $u$ with $r'_u = r'(S_h^u)$ for some $h$. So the LP completion time for $u$ is $C_u = r'(S_h^u) + p_u$. For any $q \in V$, $C_q$ is given by (7). By randomizing $\alpha \in [0,1]$, Goemans [4] derived from $A_\alpha$ its randomized version $RANDOM_\alpha$ and derandomized version $BEST_\alpha$ algorithms. Here, we need to show that for some $\rho_1$, the expected objective value with respect to $C^\alpha$ satisfies

$$
(11) \quad E[wC^\alpha] = \sum_{j \in N} w_j E[C_j^\alpha] = \sum_{u \in U} w_u(E[r^\alpha(S_h^u)] + p_u) + \sum_{q \in V} w_q E[C_q^\alpha] \le \sum_{j \in N} \rho_1 w_j C_j.
$$

So it suffices to show that for any minimal tight set $S_h$ and any preempted job $q$,

$$
(12) \quad (i)\ E[r^\alpha(S_h)] \le \rho_1 r'(S_h), \qquad (ii)\ E[C_q^\alpha] \le \rho_1 C_q.
$$

Now, suppose by induction that $E[r^\alpha(Q)] \le \rho_1 r'(Q)$ holds for all blocks $Q$ containing $S_h$ (Note that it trivially holds for $Q = N$ by definition.) We are done if we show that (i) and (ii) of (12) hold.

Consider the random choice of $\alpha \in [0,1]$ with probability distribution function $F(x)$. Let $\bar{F}(x) = 1 - F(x)$.

**Lemma 4.1** *Suppose that $\bar{F}(x) \le c(1-x)$ for all $x \in [0,1]$. Then*

1. $\sum_{j=1}^{k} a_j \bar{F}(\alpha_j) \le \frac{1}{2}c$;
2. $\sum_{j=1}^{h} a_j \bar{F}(\alpha_j) \le c\alpha_h \left(1 - \frac{1}{2}\alpha_h\right)$ *for* $h = 1, \ldots, k-1$.

**Proof:** For (1), $\sum_{j=1}^{k} a_j \bar{F}(\alpha_j) \le c\sum_{j=1}^{k} a_j(1-\alpha_j) = c\sum_{j=1}^{k}\sum_{i=j+1}^{k+1} a_j a_i \le \frac{1}{2}c$. For (2),

$$\begin{aligned}\sum_{j=1}^{h} a_j \bar{F}(\alpha_j) &\le c\sum_{j=1}^{h}\sum_{i=j+1}^{k+1} a_j a_i = c\sum_{j=1}^{h-1}\sum_{i=j+1}^{h} a_j a_i + c\sum_{j=1}^{h}\sum_{i=h+1}^{k+1} a_j a_i \\ &\le \frac{1}{2}\alpha_h^2 c + \alpha_h(1-\alpha_h)c = \alpha_h\left(1 - \frac{1}{2}\alpha_h\right)c. \quad \square\end{aligned}$$

Taking the expectations of (9), (10) and using Lemma 4.1, we obtain

$$\begin{aligned}(13)\quad E[r^\alpha(S_h)] &\le E[r^\alpha(Q)] + E[(1+\alpha)]\sum_{j=1}^{h-1} p(S_j) + p_q F(\alpha_h) + \sum_{j=1}^{h} p_q a_j \bar{F}(\alpha_j) \\ &\le E[r^\alpha(Q)] + E[(1+\alpha)]\sum_{j=1}^{h-1} p(S_j) + p_q\alpha_h\left[\frac{F(\alpha_h)}{\alpha_h} + c\left(1 - \frac{1}{2}\alpha_h\right)\right].\end{aligned}$$

$$\begin{aligned}(14)\quad E[C_q^\alpha] &\le E[r^\alpha(Q)] + p_q + \sum_{j=1}^{k} p_q a_j \bar{F}(\alpha_j) + \sum_{j=1}^{k} p(S_j) E\left[(1+\alpha)I(\alpha > \alpha_j)\right] \\ &\le E[r^\alpha(Q)] + p_q\left(1 + \frac{1}{2}c\right) + \sum_{j=1}^{k}(1-\alpha_j)p(S_j)\left(\frac{E\left[(1+\alpha)I(\alpha > \alpha_j)\right]}{1-\alpha_j}\right).\end{aligned}$$

We need to find a proper distribution function $F(x)$ over $x \in [0,1]$ such that the terms of $E[r^\alpha(S_h)]$ and $E[C_q^\alpha]$ are uniformly bounded by a factor $\rho_1$ of their respective terms of $r(S_h)$ in (6) and $C_q$ in (7). That is, find $F(x)$ with $\rho_1$ as small as possible subject to

$$\begin{aligned}(15)\quad &(i)\ \frac{F(x)}{x} + c\left(1 - \frac{1}{2}x\right) \le \rho_1,\ \forall x \in [0,1], \quad (ii)\ \frac{\bar{F}(x)}{1-x} \le c,\ \forall x \in [0,1], \\ &(iii)\ \frac{E\left[(1+\alpha)I(\alpha > x)\right]}{1-x} \le \rho_1,\ \forall x \in [0,1], \quad (iv)\quad \max\{1 + \frac{1}{2}c,\ 1 + E[\alpha]\} \le \rho_1.\end{aligned}$$

Notice that (ii) implies $c \ge 1$. So by (iv), $\rho_1 \ge 1.5$. To restrict the search for $F(x)$, we observe by (i) and (ii) that in the extreme cases, such $F(x)$ should satisfy

$$\lim_{x\to 0}\frac{F(x)}{x} + c \le \rho_1, \qquad \lim_{x\to 1}\frac{\bar{F}(x)}{1-x} \le c.$$

Further, (iii) suggests using a density function of the form $\rho_1/(1+y)$ for $y \in [x, 1]$. It appears that a good candidate for the probability density function has the form

$$(16)\quad \xi(x) = \begin{cases} A(1+x)e^{bx}, & \text{if } 0 \le x < a \\ \frac{\rho_1}{1+x}, & \text{if } a \le x \le 1. \end{cases}$$

Using the MAPLE software, we derive by properly balancing all constraints of (15) the parameters $A, \rho_1, b$, and $a$ as given in (3).

**Lemma 4.2** *If the probability density function $\xi(x)$ is given by (3), then its distribution function $F(x)$ satisfies (i) through (vi) in (15) with $c = 1.0556$, $\rho_1 = 1.7735$ and $E[\alpha] \approx 0.51012143$.*

**Proof:** Using

$$(17)\quad F(x) = \begin{cases} \frac{A}{b}\left[e^{bx}\left(1+x-\frac{1}{b}\right) - 1 + \frac{1}{b}\right], & \text{if } 0 \le x < a \\ 1 - \rho_1 \ln(\frac{2}{1+x}), & \text{if } a \le x \le 1 \end{cases}$$

with the parameters given in (3), (i), (ii) are easily verified by the MAPLE. By calculation, $E[\alpha] \approx 0.51012143$, and so (iv) holds. For (iii), notice that $1.7735/(1+x) \ge 0.7179e^{.34067}$ for all $x \le 0.454628$. So

$$E[(1+\alpha)I(\alpha > x)] = \int_x^1 (1+y)\xi(y)\,dy \le \int_x^1 (1+y)\frac{1.7735}{1+y}\,dy = 1.7735(1-x),$$

as required. □

Now, we observe that by (iii), such a choice of $F(x)$ is the best possible for $x \in [a, 1]$. For $x \in [0, a]$, the function $\widetilde{F}(x) = F(x)/x + c(1 - x/2)$ is convex with $\widetilde{F}(0) = 1.7735$, $\widetilde{F}(a) = 1.773$ and the minimum $\widetilde{F}(x) = 1.768$. Furthermore, $\widetilde{F}(x)$ is concave in $[a, 1]$ with $\widetilde{F}(0.466) = 1.7735$, and the constraint (ii) is binding at $x = 0.3189$. So, it appears that there is little hope to find other distribution functions with $\rho_1$ significantly better than 1.7735. (We tried some, but they lead no improvements.)

Using the above lemmas and the induction, we derive

**Theorem 4.3** *The $RANDOM_\alpha$ has the expected performance guarantee* $(1.7735, 1.51)$, *and the $BEST_\alpha$ has the performance guarantee* $\left(1.7735, 2 - \frac{1}{p_{max}}\right)$.

Finally, we present a family of bad instances to show that it is impossible to derive the performance guarantee $\rho_1 \le 1.5$ for approximation algorithms with this LP lower bound.

**Bad instances:** Let $N = \{q, 1, \ldots, n\}$. Let $w_q = \frac{1}{n^2}$, $p_q = n^2$, $r_q = 0$, and $w_j = \frac{2}{n^3}$, $p_j = 1$, $r_j = jn - 1$ $j = 1, \ldots, n$. It is not hard to verify that $N$ forms a block with job $q$ preempted $n$ times. By (7), we derive its optimal LP value

$$wC = \sum_{j=1}^{n} w_jC_j + w_qC_q = 1 + \frac{1}{n} + 1 + \frac{1}{n^2}\sum_{j=1}^{n}\left[1 - \frac{j(n-1)}{n^2}\right] \longrightarrow 2, \text{ as } n \to \infty.$$

Let $C^h$ be the schedule of sequence $q$ at $h$ for $h = 1, \ldots, n+1$. Then for $j \in N$,

$$C_j^h = \begin{cases} jn & \text{if } \ 1 \leq j \leq h-1 \\ (h-1)n + n^2 & \text{if } \ j = q \\ (h-1)n + n^2 + 1 + j - h & \text{if } \ h \leq j \leq n. \end{cases}$$

By calculation, one can show that

$$wC^h = 3 + \epsilon(n), \quad \text{where } 0 \leq \epsilon(n) \to 0 \text{ as } n \to \infty,$$

with asymptotically optimal schedules such as sequencing $q$ either first or last.

## 5 Concluding remarks

We presented an $O(n \log n)$ combinatorial algorithm for finding feasible schedules for the problem $1|r_j|\sum_j w_j C_j$ with the bicriteria guarantee. We establish the performance guarantee by fully exploiting the properties of optimal solutions of the known tight LP formulations and some randomization technique. We also derive the improved performance guarantees for the Goemans's algorithms. We believe that this type of LP-based approach has potential to design efficient and effective approximation algorithms for more complex scheduling problems.


### Acknowledgments

The author would like to thank Laurence Wolsey for helpful discussions and comments on this paper, Michel Goemans and Andeas Schulz for communicating to the author with their latest results, and Sanjeev Mahajan for reading the paper.


## References


[1] S. Chakrabarti, C.A.Phillips, A.S.Schulz, D.B.Shmoys, C.Stein and J.Wein, "Improved scheduling algorithms for minsum criteria", to appear in Springer CNCS, Proc. of ICALP 1996.

[2] M. Dyer and L. Wolsey, "Formulating the single machine sequencing problem with release dates as a mixed integer program", *Discrete Applied Mathematics 26* 255-270 (1990).

[3] M.X.Goemans, "A supermodular relaxation for scheduling with release dates", *Proceedings of the Fifth Integer Programming and Combinatorial Optimization Conference*, W.H.Cunningham, S.T.McCormick and M.Queyranne, eds., LNCS 1084, Springer, 288-300 (1996).

[4] M.X.Goemans, "Improved approximation algorithms for scheduling with release dates", July 1, 1996.

**Appendix A:**

For notational convenience, let $PR$ denote a priority queue with keys corresponding to the $\frac{w_j}{p_j}$ ratios. Let $PR(1)$ be the top element (with the largest ratio), and $PR.key(1)$ as the key for the top element. Let **Insert**$(j, PR)$ denote the operation of inserting job $j$ into the priority queue, and let **Delete**$PR(1)$ be the operation of deleting the top element from the queue. For each $j \in N$, let $p'_j$ denote the amount of remaining work in the preemptive scheduling process, and let $pred(j, T)$ be its immediate predecessor in $T$ (Note that each $j$ has the unique predecessor by construction.) We use **Add**$(vj, T)$ to denote the operation of appending a new node $j$ to $v \in T$ along with link $(v, j)$ to $T$. We also introduce a dummy job $d$ to indicate if all preempted jobs are completed. As input, we assume that $\frac{w_d}{p_d} = -1$, and $r_{n+1} = r_n + p(N)$ (to ensure the completion of the remaining jobs in $PR$.)

**D-Tree** ( for constructing a decomposition tree )
**Input:** $w, p, r$; **Output:** the decomposition tree $T$ and $r'$;
**Order** release dates as $r_1 \leq \cdots \leq r_n$; **Insert**$(d, PR)$;
$V(T) \leftarrow \{d\}$; $E(T) \leftarrow \emptyset$; $v \leftarrow d$;
**for** $j = 1$ **to** $n$ **do** $p'_j \leftarrow p_j$, $r'_j \leftarrow r_j$;

**for** $j = 1$ **to** $n$ **do**
**begin** $t \leftarrow r_j$; (This statement is necessary if the machine has been idle before.)
(i) **if** $\frac{w_j}{p_j} \leq \frac{w_v}{p_v}$ **then** **Insert**$(j, PR)$ **else** **Add**$(vj, T)$ , $v \leftarrow j$;
(ii) $t_{new} \leftarrow \min\{t + p'_v, r_{j+1}\}$ ; $p'_v \leftarrow t + p'_v - t_{new}$; $t \leftarrow t_{new}$;
**if** $p'_v = 0$ **then**
**begin** $u \leftarrow prec(v, T)$;
**if** $PR.key(1) \leq \frac{w_u}{p_u}$ **then** $v \leftarrow u$
**else** $v \leftarrow PR(1)$, **Add**$(uv, T)$, **Delete**$PR(1)$, $r'_v \leftarrow t$;
**if** $v \neq d$ **then** **return to** (ii)
**end**
**Stop** **if** $T$ contains $n$ nodes.
**end**

We use the following example from Dyer and Wolsey [2] to demonstrate how **D-Tree** works.

| $j$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $r_j$ | 0 | 1 | 3 | 8 | 8 | 11 | 12 | 16 | 23 | 27 |
| $w_j$ | 5 | 2 | 2 | 6 | 1 | 20 | 4 | 12 | 5 | 10 |
| $p_j$ | 7 | 4 | 5 | 3 | 8 | 3 | 1 | 5 | 6 | 9 |

The constructed preemptive schedule is

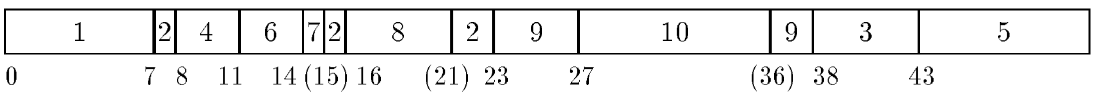

**Figure 1:** The preemptive schedule

The numbers in the boxes are indices of jobs. Other numbers without brackets are the set $r'$ of release dates. The following figure exhibits the decomposition tree.

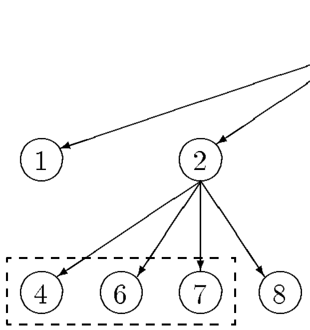


**Figure 2:** The decomposition tree

In this example, there are 8 unpreempted jobs, i.e., eight leaves in $T$. Two preempted jobs 2 and 9 correspond to the two internal nodes in $T$ with their associated blocks $\{2,4,6,7,8\}$ and $\{9,10\}$.

To describe the algorithm, we need more notation for convenience. Let $QH$ denote the priority queue for holding nodes with keys corresponding to their heights. ($QH$ can be created in $O(n)$ time.) The top element $QH(1)$ contains the node with the lowest height among all nodes in $QH$.

$GP(\beta)$ (The Greedy-Position )
**Input:** $w, p, r; \beta$ **Output:** the optimal LP solution $C$ and the processing sequence;

**Call D-Tree**; **Create** $QH$;
**while** $QH.key(1) = 0$ **do begin** $l \leftarrow QH(1)$, $C_l \leftarrow r'_l + p_l$, **Delete**$QH(1)$ **end**;

**while** $QH(1) \neq d$ **to**
**begin** $q \leftarrow QH(1)$; **Delete**$QH(1)$;
 **S1:Construct** the block $(\{q\}, S_1, \dots, S_k)$ with the *maximal* tight set $S_i$.
 **S2:**$r'(Q) \leftarrow r'_q$; $p(Q) \leftarrow \sum_{j=1}^{k} p(S_j) + p_q$; $w(Q) \leftarrow \sum_{j=1}^{k} w(S_j) + w_q$;
 **S3:**$a_1 \leftarrow [r'(S_1) - r'_q]/p_q$; $a_{k+1} \leftarrow [r'_q + p(Q) - r'(S_k) - p(S_k)]/p_q$;
  **for** $j = 2$ **to** $k$ **do** $a_j \leftarrow [r'(S_j) - r'(S_{j-1}) - p(S_{j-1})]/p_q$;
 **S4:**$Psum_1 \leftarrow p(S_1)$; $t_q \leftarrow 0$;
  **for** $j = 2$ **to** $k$ **do** $Psum_j \leftarrow Psum_{j-1} + p(S_j)$;
  **for** $j = 1$ **to** $k$ **do** $t_q \leftarrow t_q + a_{j+1} Psum_j$;
  $C_q \leftarrow r'_q + p_q + t_q$;
 **S5:**$h(\beta) \leftarrow \min\{h : \sum_{j=1}^{h} a_j > \beta\}$; $W_1 \leftarrow w(Q) - w_q$; (Compute the function $B(h,\beta)$.)
  **for** $h = 2$ **to** $h(\beta)$ **do** $W_j \leftarrow W_{j-1} - w(S_{j-1})$;
  $B(1) \leftarrow p_q w(Q)$;

**for** $h = 2$ **to** $h(\beta)$ **do**

$B(h, \beta) = B(h-1) + p_q\left[a_{h-1}W_{h-1} - w(S_{h-1})\right] + w_q\left[p_q a_{h-1} + (1+\beta)p(S_{h-1})\right];$

**S6:** Insert job $q$ at the position $h^*$ with the minimum $B(h, \beta)$; **Record** the position $h^*$;

**end**

In **S2**, we compute $r(Q)$, $w(Q)$ and $p(Q)$ for later use by its predecessor.

We use the previous example again to demonstrate how the algorithm works.

**Example:** (Assume that we choose $\beta > \frac{2}{3}$.)

This example also explains **S1**. Here we have some $a_h = 0$. This means that $q$ will be sequenced at $h$ with probability 0. So there are only three possible positions to consider for the block $Q = \{2, 4, 6, 7, 8\}$. That is, $2 \to \{4, 6, 7\} \to \{8\}$, $\{4, 6, 7\} \to 2 \to \{8\}$ and $\{4, 6, 7\} \to \{8\} \to 2$. So the constructed $Q = \{\{2\}, \{4, 6, 7\}, \{8\}\}$ in **S1**. We have $a_1 = a_2 = \frac{1}{4}$ and $a_3 = \frac{1}{2}$, and $C_2 = 7 + 4 + \frac{1}{4}(7) + \frac{1}{2}(7+5) = 18.75$.

$B(1) = 4(30 + 12 + 2) = 176$
$B(2) = 176 + 4[\frac{1}{4}(30+12) - 30] + 2[\frac{1}{4}(4) + (1+\beta)(7)] = 114 + 14\beta$
$B(3) = B(2) + 4[\frac{1}{4}(12) - 12] + 2[\frac{1}{4}(4) + (1+\beta)(5)] = 90 + 24\beta.$

So $h^* = 3$ is the best position. For the block $Q = (\{9\}, \{10\})$, we have $a_1 = \frac{2}{3}$ and $a_2 = \frac{1}{3}$, and $C_9 = 23 + 6 + \frac{1}{3}(9) = 32$.

$B(1) = 6(10 + 5) = 90$
$B(2) = 90 + 6[\frac{2}{3}(10) - 10] + 5[\frac{2}{3}(6) + (1+\beta)(9)] = 135 + 45\beta.$

So $h^* = 1$ is the best position for job 9. Hence, the job processing sequence obtained from $GP(\beta)$ is $(1, 4, 6, 7, 8, 2, 9, 10, 3, 5)$, which happens to be the optimal sequence as shown in [2].

**Appendix B:**

The following identity will be implicitly used in the proofs.

(18) $g(S_1 \cup S_2) + g(S_1 \cap S_2) = g(S_1) + g(S_2) + p(S_1 \setminus S_2)p(S_2 \setminus S_1)$ for all $S_1,\ S_2 \subseteq N$.

For simlicity, let $p * C(S) = \sum_{j \in S} p_j C_j$.

**Proof of Lemma 2.1:** From $r(S_1) \leq r(S_2)$, we obtain $r(S_1) = r(S_1 \cup S_2)$. Further, using $-r(S_1 \cap S_2) \leq -r(S_2)$, we have

$$\begin{aligned}
& r(S_1)p(S_1 \cup S_2) + g(S_1 \cup S_2) = r(S_1 \cup S_2)p(S_1 \cup S_2) + g(S_1 \cup S_2) \\
\leq \quad & p * C(S_1 \cup S_2) = p * C(S_1) + p * C(S_2) - p * C(S_1 \cap S_2) \\
\leq \quad & r(S_1)p(S_1) + g(S_1) + r(S_2)p(S_2) + g(S_2) \\
& -r(S_1 \cap S_2)p(S_1 \cap S_2) - g(S_1 \cap S_2) \\
\leq \quad & r(S_1)p(S_1) + g(S_1) + r(S_2)p(S_2) + g(S_2) \\
& -r(S_2)p(S_1 \cap S_2) - g(S_1 \cap S_2) \\
= \quad & r(S_1)p(S_1) + r(S_2)p(S_2 \setminus S_1) + g(S_1 \cup S_2) - p(S_1 \setminus S_2)p(S_2 \setminus S_1).
\end{aligned}$$

The last equality follows from (18). Simplifying, we obtain

$$r(S_1)p(S_2 \setminus S_1) + p(S_1 \setminus S_2)p(S_2 \setminus S_1) \leq r(S_2)p(S_2 \setminus S_1).$$

Dividing by $P(S_2 \setminus S_1) > 0$, the proof is complete.

Notice that the above proof is valid in the case of $S_1 \cap S_2 = \emptyset$. $\square$

**Proof of Lemma 2.2:** It follows from Lemma 2.1 by induction on $k$ and simple calculations. Next, we prove a slightly extended version of **Theorem 2.3**.

**Theorem 5.1** *Let $S_1$ and $S_2$ be any two tight sets that properly intersect (that is, all $S_1 \setminus S_2$, $S_2 \setminus S_1$ and $S_1 \cap S_2$ are nonempty), and assume $r(S_1) \leq r(S_2)$. Then the following statements hold:*

1. $r(S_1 \setminus S_2) = r(S_1) = r(S_1 \cup S_2)$;
2. $r(S_2) = r(S_1) + p(S_1 \setminus S_2)$, *and* $r(S_2) = r(S_1 \cap S_2)$;
3. *All sets $S_1 \setminus S_2$, $S_1 \cap S_2$, and $S_1 \cup S_2$ are tight.*

**Proof:** By tightness of $S_1$ and monotonicity of the function $r$, we have

$$\begin{aligned}
r(S_1)p(S_1) + g(S_1) \quad &= \quad p * C(S_1) = p * C(S_1 \setminus S_2) + p * C(S_1 \cap S_2) \\
&\geq \quad r(S_1 \setminus S_2)p(S_1 \setminus S_2) + g(S_1 \setminus S_2) \\
&\quad\quad +r(S_1 \cap S_2)p(S_1 \cap S_2) + g(S_1 \cap S_2) \\
&= \quad r(S_1 \setminus S_2)p(S_1 \setminus S_2) + r(S_1 \cap S_2)p(S_1 \cap S_2) \\
&\quad\quad +g(S_1) - p(S_1 \setminus S_2)p(S_1 \cap S_2) \\
&\geq \quad r(S_1)p(S_1 \setminus S_2) + r(S_2)p(S_1 \cap S_2) \\
&\quad\quad +g(S_1) - p(S_1 \setminus S_2)p(S_1 \cap S_2).
\end{aligned} \tag{19}$$

Further, by Lemma 2.1 (substituting $r(S_2)$), the above is simplified as follows.

$$
\begin{aligned}
(20)\quad r(S_1)p(S_1) &\geq r(S_1)p(S_1 \setminus S_2) - p(S_1 \setminus S_2)p(S_1 \cap S_2) + r(S_2)p(S_1 \cap S_2) \\
&\geq r(S_1)p(S_1 \setminus S_2) - p(S_1 \setminus S_2)p(S_1 \cap S_2) \\
&\quad + [r(S_1) + p(S_1 \setminus S_2)]\, p(S_1 \cap S_2) \\
&= r(S_1)p(S_1 \setminus S_2) + r(S_1)p(S_1 \cap S_2) = r(S_1)p(S_1).
\end{aligned}
$$

So equalities must hold throughout in (19) and (20). Notice that in (20), we have used those inequalities involving the function $r$ and the shifted parallel inequalities with respect to $S_1 \setminus S_2$ and $S_1 \cap S_2$. This proves the first two statements and part of statement 3. Now, since $r(S_2) = r(S_1) + p(S_1 \setminus S_2) = r(S_1 \setminus S_2) + p(S_1 \setminus S_2)$, and both $S_1 \setminus S_2$ and $S_2$ are tight, by Lemma 2.1, $S_1 \cup S_2$ is also tight. The proof is complete. □

**Proof of expression (7)**: Since $Q$ is tight, by (6), we have

$$
\begin{aligned}
f(Q) &\doteq r'(Q)p(Q) + g(Q) = r'(Q)p(Q) + p_q^2 + p_q \sum_{i=1}^{k} p(S_i) + g\left(\bigcup_{i=1}^{k} S_i\right) \\
&= p_q C_q + \sum_{i=1}^{k} p * C(S_i) = p_q C_q + \sum_{i=1}^{k} [r'(S_i)p(S_i) + g(S_i)] \\
&= p_q C_q + \sum_{i=1}^{k} g(S_i) + \sum_{i=1}^{k} p(S_i) \left[r'(Q) + p_q \sum_{j=1}^{i} a_j + \sum_{j=1}^{i-1} p(S_j)\right] \\
&= p_q C_q + g\left(\bigcup_{i=1}^{k} S_i\right) + \sum_{i=1}^{k} p(S_i) \left[r'(Q) + p_q \sum_{j=1}^{i} a_j\right].
\end{aligned}
$$

Comparing the third and the last expressions above, we derive $C_q = t_q + p_q$, where $t_q$ is the *LP start time* for job $q$ given by

$$
(21)\quad t_q = r'(Q) + \sum_{i=1}^{k} p(S_i) \left[1 - \sum_{j=1}^{i} a_j\right] = r'(Q) + \sum_{j=2}^{k+1} a_j \sum_{i=1}^{j-1} p(S_i). \qquad \square
$$

**Proof of Proposition 3.4:** Observe first that if the job $q$ is inserted at $h$, then any block $S_j$ with $j > h$ can start immediately after completion of the last job in $S_{j-1}$ (no need to delay $a_j p_q$ units of time). Next, note that by the induction hypothesis, the jobs in $S_i$ can be scheduled between $r'(S_i) + (1+\beta)p(S_i)$ for $i = 1, \ldots, k$. Therefore, by the choice of $h(\beta)$, the completion time for the last job in $H^h$ is at most

$$
p_q \sum_{i=1}^{h-1} a_i + (1+\beta) \sum_{j=1}^{k} p(S_j) + p_q \leq p_q \beta + (1+\beta) \sum_{j=1}^{k} p(S_j) + p_q = (1+\beta)p(Q),
$$

as required. □

**Proof of Proposition 3.5:** Since $Q$ is a block, and the processing times and release dates are integers, we have

$$
a_{k+1} p_q = r'(Q) + p(Q) - r'(S_k) - p(S_k) \geq 1 \Longrightarrow a_{k+1} \geq \frac{1}{p_q} \geq \frac{1}{p_{max}}.
$$

It follows that

$$\begin{aligned}
k+1 &\geq h(\beta) = \min\left\{h : \sum_{j=1}^{h} a_j > \beta\right\} = \min\left\{h : \sum_{j=1}^{h} a_j > 1 - \frac{1}{p_{max}}\right\} \\
&\geq \min\left\{h : \sum_{j=1}^{h} a_j > 1 - a_{k+1}\right\} = k+1.
\end{aligned}$$

Hence $\beta^* = \sum_{j=1}^{h(\beta)} a_j = 1$. The statement 2 follows from Proposition 3.4. $\Box$

**Proof of Lemma 3.3:** Write $B(h,\beta) = X(h,\beta) + Y(h,\beta)$, which correspond to two terms of $B(h,\beta)$.

$$\begin{aligned}
(22)\quad E[X(h,\beta)] &= p_q\left[\sum_{h=1}^{h(\beta)} \frac{a_h}{\beta^*} \sum_{j=1}^{h-1} a_j \sum_{i=j}^{k} w(S_i) + \sum_{h=1}^{h(\beta)} \frac{a_h}{\beta^*} \sum_{i=h}^{k} w(S_i)\right] \\
&= p_q\left[\sum_{j=1}^{h(\beta)-1} a_j \sum_{i=j}^{k} w(S_i) \sum_{h=j+1}^{h(\beta)} \frac{a_h}{\beta^*} + \sum_{j=1}^{h(\beta)} \frac{a_j}{\beta^*} \sum_{i=j}^{k} w(S_i)\right] \\
&= p_q \sum_{j=1}^{h(\beta)-1} a_j \sum_{i=j}^{k} w(S_i) \left[\sum_{h=j+1}^{h(\beta)} \frac{a_h}{\beta^*} + \frac{1}{\beta^*}\right] \\
&\leq \left(1 + \frac{1}{\beta^*}\right) p_q \sum_{j=1}^{k} a_j \sum_{i=j}^{k} w(S_i).
\end{aligned}$$

$$\begin{aligned}
(23)\quad E[Y(h,\beta)] &= w_q\left[p_q \sum_{h=1}^{h(\beta)} \frac{a_h}{\beta^*}\left(1 + \sum_{j=1}^{h-1} a_j\right) + (1+\beta) \sum_{h=1}^{h(\beta)} \frac{a_h}{\beta^*} \sum_{i=1}^{h-1} p(S_i)\right] \\
&\leq w_q\left[p_q\left(1 + \frac{\beta^*}{2}\right) + \left(1 + \frac{1}{\beta^*}\right) \sum_{j=2}^{k+1} a_j \sum_{i=1}^{j-1} p(S_i)\right]. \quad \Box
\end{aligned}$$

**Proof of Theorem** 3.1: For $i = 1, \ldots, k$, let $C^i$ denote the optimal LP solution to the problem

$P(w, p, \tilde{r}, S_i)$, where $\tilde{r}_j = r'_j - r'(S_i)$ for $j \in S_i$. By (6) and (7), the optimal LP objective value can now be expressed as (note $r'(Q) = 0$.)

$$\begin{aligned}
(24)\quad wC &= \sum_{i=1}^{k} \sum_{s \in S_i} w_s C_s + w_q C_q = \sum_{i=1}^{k} \left[w(S_i) r'(S_i) + \sum_{s \in S_i} w_s C_s^i\right] + w_q C_q \\
&= \sum_{i=1}^{k} \sum_{s \in S_i} w_s C_s^i + \sum_{i=1}^{k} w(S_i) \left[\sum_{j=1}^{i} a_j p_q + \sum_{j=1}^{i-1} p(S_j)\right] + w_q C_q \\
&= \sum_{i=1}^{k} \sum_{s \in S_i} w_s C_s^i + \sum_{j=1}^{k-1} p(S_j) \sum_{i=j+1}^{k} w(S_j)
\end{aligned}$$

$$+p_q \sum_{j=1}^{k} a_j \sum_{i=j}^{k} w(S_i) + w_q \left[ p_q + \sum_{j=2}^{k+1} a_j \sum_{i=1}^{j-1} p(S_i) \right].$$

For each $h = 1, \ldots, k+1$, we consider a feasible schedule $H^h$ for $p(w, p, r', Q)$ as follows. For $i = 1, \ldots, k$, let $\hat{C}^i$ be the feasible schedule for $P(w, p, \widetilde{r}, S_i)$ obtained from the algorithm. When we sequence job $q$ at position $h$, the resulting schedule has the form $H^h = (F^1, \ldots, F^{h-1}, \hat{C}_q, F^h, \ldots, F^k)$, where $\hat{C}_q = p_q + p_q \sum_{j=1}^{h-1} a_j + (1+\beta) \sum_{j=1}^{h-1} p(S_j)$ and for $i = 1, \ldots, k$,

$$(25) \quad F_s^i = \hat{C}_s^i + I(1 \le i < h) p_q \sum_{j=1}^{i} a_j + I(h \le i \le k) \left( p_q + p_q \sum_{j=1}^{h-1} a_j \right) + (1+\beta) \sum_{j=1}^{i-1} p(S_j).$$

Note that $I(.)$ is an indicator function. The objective value of $H^h$ is

$$wH^h = w_q \hat{C}_q + \sum_{i=1}^{k} \sum_{s \in S_i} w_s F_s^i.$$

By the induction hypothesis, we have for each $i$ that $\sum_{s \in S_i} w_s \hat{C}_s^i \le \rho \sum_{s \in S_i} w_s C_s^i$, where $\rho = 1 + \frac{1}{\beta}$, and therefore by (25) each term of $wH^h$ is bounded above as follows.

$$(26) \quad w_q \hat{C}_q = w_q \left[ p_q + p_q \sum_{j=1}^{h-1} a_j + (1+\beta) \sum_{j=1}^{h-1} p(S_j) \right];$$

for $1 \le i < h$,

$$(27) \quad \sum_{s \in S_i} w_s F_s^i \le \rho \sum_{s \in S_i} w_s C_s^i + w(S_i) \left[ p_q \sum_{j=1}^{i} a_j + (1+\beta) \sum_{j=1}^{i-1} p(S_j) \right],$$

and for $h \le i \le k$,

$$(28) \quad \sum_{s \in S_i} w_s F_s^i \le \rho \sum_{s \in S_i} w_s C_s^i + w(S_i) \left[ p_q + p_q \sum_{j=1}^{h-1} a_j + (1+\beta) \sum_{j=1}^{i-1} p(S_j) \right].$$

Summing over the above and exchanging some summations, we obtain an upper bound $F(h)$ for $wH^h$:

$$\begin{aligned}
(29) \quad wH^h \quad \le \quad & \rho \sum_{i=1}^{k} \sum_{s \in S_i} w_s C_s^i + (1+\beta) \sum_{j=1}^{k-1} p(S_j) \sum_{i=j+1}^{k} w(S_i) \\
& + p_q \left[ \sum_{j=1}^{h-1} a_j \sum_{i=j}^{k} w(S_i) + \sum_{i=h}^{k} w(S_i) \right] \\
& + w_q \left[ p_q \left( 1 + \sum_{j=1}^{h-1} a_j \right) + (1+\beta) \sum_{j=1}^{h-1} p(S_j) \right] \\
= \quad & \rho \sum_{i=1}^{k} \sum_{s \in S_i} w_s C_s^i + (1+\beta) \sum_{j=1}^{k-1} p(S_j) \sum_{i=j+1}^{k} w(S_i) + B(h, \beta) \equiv F(h).
\end{aligned}$$

By comparing $F(h)$ with $wC$ in (24), we observe that the first two terms of $F(h)$ are independent of $h$ and are at most $\rho$ times the corresponding terms of $wC$. Consider the choice of $h$ with probability $\frac{a_h}{\beta^*}$. Since we choose $h^*$ with the minimum $B(h, \beta)$, it follows from Lemma 3.3 that

$$wH^{h^*} \quad \leq \quad \rho \sum_{i=1}^{k} \sum_{s \in S_i} w_s C_s^i + (1+\beta) \sum_{j=1}^{k-1} p(S_j) \sum_{i=j+1}^{k} w(S_i) + E[B(h, \beta)] \leq \left(1 + \frac{1}{\beta}\right) wC,$$

as required. □

**Proof of Theorem** 3.2: By Proposition 3.5 and the induction hypothesis, $F(h) - B(h)$ is at most $\left(2 - \frac{1}{p_{max}}\right)$ times the first two terms of $wC$ (note that here we set $\rho = 2 - \frac{1}{p_{max}}$.) Further, observe that since the processing times are integers, by definition, $a_1 \geq \frac{1}{p_{max}}$. So for $j \geq 1$,

$$\sum_{h=j+1}^{k+1} a_h = 1 - \sum_{i=1}^{j} a_j \leq 1 - \frac{1}{p_{max}}.$$

Thus, from (22) with $\beta = 1 - \frac{1}{p_{max}}$, we derive

$$E[X(h, \beta)] = p_q \sum_{j=1}^{h(\beta)} a_j \sum_{i=j}^{k} w(S_i) \left[ \sum_{h=j+1}^{h(\beta)} \frac{a_h}{\beta^*} + \frac{1}{\beta^*} \right] \leq \left(2 - \frac{1}{p_{max}}\right) p_q \sum_{j=1}^{h(\beta)} a_j \sum_{i=j}^{k} w(S_i).$$

Similarly, from (23),

$$\begin{aligned} E[Y(h, \beta)] \quad &= \quad w_q \left[ p_q \sum_{h=1}^{h(\beta)} \frac{a_h}{\beta^*} \left(1 + \sum_{j=1}^{h-1} a_j\right) + (1+\beta) \sum_{h=1}^{h(\beta)} \frac{a_h}{\beta^*} \sum_{i=1}^{h-1} p(S_i) \right] \\ &\leq \quad w_q \left[ p_q \left(1 + \frac{\beta^*}{2}\right) + \left(1 + 1 - \frac{1}{p_{max}}\right) \sum_{j=2}^{k+1} a_j \sum_{i=1}^{j-1} p(S_i) \right]. \end{aligned}$$

Hence, for all $p_{max} \geq 2$, we have

$$wH^{h^*} \leq E[F(h)] \leq \left(2 - \frac{1}{p_{max}}\right) wC,$$

as required. If $p_{max} = 1$, by the assumption of integer processing times and release dates, the optimal preemptive schedule is also an optimal nonpreemptive schedule. So the performance guarantee still holds. □