On the Optimality of the Earliest Due Date Rule in Stochastic Scheduling and in Queueing

Richard Bryant[a], Peter Lakner[a], Michael Pinedo[a,*]

[a] *Department of Technology, Operations, and Statistics, Stern Business School, New York University, New York, NY 10012*

Abstract

We consider an environment with m servers in parallel and n jobs. The n jobs have i.i.d. exponentially distributed processing requirements. The servers operate at different speeds and preemptions are allowed. The jobs have random release dates that are not known in advance, but whenever a job is released, its due date is fixed and becomes known to the scheduler. We first consider three stochastic scheduling problems with three due date related objective functions and consider variations of the Earliest Due Date (EDD) rule, including the preemptive policy that at any point in time assigns the job with the Earliest Due Date to the Fastest Server (*EDD-FS*). Our optimality results turn out to be examples of stochastic scheduling problems that have relatively simple priority rules that are optimal while their deterministic counterparts do not allow such priority rules to be optimal. We furthermore extend our results to include a priority queueing model with m exponential servers that operate at different speeds and preemptions being allowed. The jobs arrive according to an arbitrary renewal process, and we assume that the utilization factor of the system is less than 1. Upon a job's arrival the time difference between its due date and arrival date is set by the draw of a random variable from a given distribution. At a job's arrival date, its due date is immediately fixed and known while its actual processing time only becomes known upon its service completion. We show that for this priority queueing model the preemptive *EDD-FS* rule also minimizes the limit of several due date related objective functions as the number of jobs converges to infinity.

Keywords: Scheduling, Multi-Server Queueing, Parallel machine scheduling, Due Date Related Objective Functions, EDD rule

*Corresponding author

Email addresses: rmb681@stern.nyu.edu (Richard Bryant), plakner@stern.nyu.edu (Peter Lakner), mpinedo@stern.nyu.edu (Michael Pinedo)

1. Introduction

Consider m servers in parallel and n jobs. The processing requirements of the n jobs are independent and exponentially distributed with the same mean. Job j is released at a random time R_j, and its due date D_j is a random variable that is fixed and becomes known to the scheduler at the arrival time of job j. When the random due date D_j of job j is fixed and known we denote it by d_j. We require that the vector of random variables $(R_1, \ldots, R_n, D_1, \ldots, D_n)$ is independent of the servers. We shall not assume that $D_j > R_j$, though in practice most of the times this is the case. The servers operate at different speeds. If a job is processed on server i its remaining processing time on server i is exponentially distributed with rate μ_i. Without loss of generality we assume that $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_m$. That is, server 1 operates at the highest speed, server 2 at the second highest, and so on. Preemptions are allowed, i.e., we are allowed at any point in time to take an unfinished job from one server and either put it on another server or put it back in the waiting line and resume its processing at some later point in time. Let C_j denote the completion time of job j. This completion time clearly depends on the scheduling policy. Let $L_j = C_j - d_j$ denote the lateness of job j and let $T_j = \max(C_j - d_j, 0)$ denote its tardiness. Let $U_j = 1$ if job j is completed after its due date, i.e., it is tardy, and 0 otherwise. We are interested in three different objective functions to minimize, namely $E(L_{\max}) = E(\max(L_1, \ldots, L_n))$, $E(\sum T_j)$, and $E(\sum U_j)$.

In the case of due date related objective functions, a well-known rule in scheduling theory is the so-called Earliest Due Date first (EDD) rule which prioritizes the jobs according to their due dates. This rule is commonly used in many different types of environments, including production scheduling settings and real time computing systems, see Pinedo [16]. In the computer science literature this rule is often also referred to as the Earliest Deadline First (EDF) rule, see Buttazzo [4] and Baruah et al. [2]. In this paper we establish some theoretical properties of generalizations of the EDD rule when used in stochastic environments.

We consider two different variations of the EDD rule. According to the preemptive *Earliest Due Date on the Fastest Server (EDD-FS)* rule the scheduler assigns at any point in time, among the jobs available for processing, the job with the *Earliest Due Date to the Fastest Server*, the job with the second *Earliest Due Date to the second Fastest Server*, and so on. Preemptions are allowed and take place only when either one of the following two types of events occur. First, at the arrival or release of a new job that has a due date that is earlier than the due date of a job currently being processed. When this occurs, the job that just arrived is put on one of the servers, and one or more jobs that are currently being processed are preempted and reassigned to slower

servers, or possibly one of those is taken off the machine and put into the queue. Second, when a job on one of the servers has completed its processing and the server is freed. The jobs that were assigned to slower servers, now move up to a faster server; the job with the earliest due date among the waiting jobs is now assigned to the slowest server, provided that there is a job waiting.

According to the preemptive *Earliest Due Date on the Fastest Server with Postponements (EDD-FS-P)* rule the scheduler assigns at any point in time, among the jobs available for processing of which the due dates have not yet occurred, the job with the *Earliest Due Date to the Fastest Server*, the job with the second *Earliest Due Date to the second Fastest Server*, and so on. However, if the due date of a job has been reached and the job has not yet been completed, then the job is taken off the server, its priority level is reduced to zero, and its processing will be resumed at some later point in time. Jobs that have not yet been completed and have not yet reached their due dates have a higher priority than jobs that are already past their due dates. A job past its due date can only resume its processing when there are fewer than m jobs in the system that have not yet been completed and have not yet reached their due dates. The jobs that have not yet reached their due date always have a higher priority than jobs that have already reached their due date. Jobs that have not yet reached their due date will therefore always be assigned to faster servers than jobs that have already reached their due dates.

In this paper we will show that the preemptive *EDD-FS* rule minimizes the objectives $E(L_{\max})$ and $E(\sum T_j)$ and that the preemptive *EDD-FS-P* rule minimizes $E(\sum U_j)$. Both rules operate in such a way that at any point in time they assign the maximum number of available jobs to the m servers, i.e., the number of jobs assigned to the m servers is the minimum of m and the number of jobs available for processing. Furthermore, if the number of jobs available for processing is less than m, then only the fastest servers are utilized while the slower servers are kept idle.

We subsequently consider a priority queueing system. We assume the same server environment with m exponential servers that operate at different speeds and jobs arrive at the system according to an arbitrary renewal process. Such a system is similar to a queue of the $GI/M/c$ type with c heterogeneous servers. (Note that in our notation the number of servers m is equivalent to the number of servers c that is often used in queueing.) We extend the classical $GI/M/c$ model by assuming that $(D_j - R_j,\ j \geq 1)$ form an i.i.d. sequence of random variables, and by assuming that the random due date D_j becomes known at the random arrival time R_j of job j. So at the arrival of a job its random due date is set equal to d_j and made known to the scheduler; however, its actual processing time only becomes known upon its completion of service. In this priority

queueing model we refer to a time interval $[s,t)$ as a *cycle* if at time s the number of jobs in the system goes up from 0 to 1, and time t is the first time after s such that the number of jobs in the system goes down from 1 to 0. We are interested in minimizing the limit of the average value of the various objective functions per cycle, as the number of cycles goes to infinity. We show that the infinite versions of the *EDD-FS* and *EDD-FS-P* policies minimize these limits, the first in the case of the total tardiness, and the second in the case of the number of late jobs.

The deterministic counterparts of the three stochastic scheduling models described above have received a significant amount of attention in the deterministic scheduling literature and the real-time computer systems literature. (Note that in the deterministic scheduling literature a server is typically referred to as a machine and in the computer science literature a server is typically referred to as a processor.) That segment of the deterministic scheduling literature that focuses on models with jobs having equal processing requirements in a parallel server environment is extensive; see, for example, Baptiste et al. [1], Leung and Young [10], and Lushchakova [13]. The problem of minimizing $L_{\max}$ in a parallel server environment with all n jobs having the same processing requirement p (i.e., $p_j = p$ for $j = 1,\ldots,n$) and the servers having different speeds is in the deterministic scheduling literature referred to as $Qm \mid prmt, p_j = p \mid L_{\max}$. This problem can be solved in polynomial time, but the solution is not just a simple preemptive priority rule like *EDD-FS*. It can only be solved through a network flow algorithm or a Linear Programming formulation, see Lawler and Labetoulle [8] and Martel [14]. The deterministic problem with objective function $\sum T_j$, i.e., $Qm \mid prmt, p_j = p \mid \sum T_j$, is from a complexity point of view still open. It is not known whether there exists a polynomial time algorithm or whether the problem is NP-Hard. However, a simple priority rule that would guarantee optimality at all times does not seem to exist. The problem with the objective function $\sum U_j$ and the m servers having the same speed, i.e., $Pm \mid r_j, prmt, p_j = p \mid \sum U_j$ is also still open; it is not known whether or not this problem is polynomial time solvable. The same can be said when the machines have different speeds; in this case also, it is not known whether or not the problem is polynomial time solvable.

A fair amount of research has been done on stochastic scheduling models and priority queueing models with jobs that have exponentially distributed processing requirements. It has often been found that scheduling problems with exponentially distributed processing times allow for simple and elegant priority rules to be optimal, while their deterministic counterparts do not allow such simple priority rules to be optimal. For an overview of such stochastic scheduling problems and comparisons to their deterministic counterparts, see Pinedo [16]. For example, Pinedo and Reed

[15] considered an environment with m servers in parallel that operate at different speeds and with jobs that have exponentially distributed processing requirements. They proved the optimality of a composite priority rule, which they referred to as the Least Flexible Job first on the Fastest Server (LFJ-FS), for several objective functions which were not due date related. However, it has to be noted that it is not always true that the deterministic version of a scheduling problem is more complicated and harder than its stochastic counterpart with exponentially distributed processing times; there are a few examples where the opposite is true, see Pinedo [16].

Queueing models of the $GI/M/c$ type have received a significant amount of attention in the literature, including models with heterogeneous servers, i.e., with servers that operate at different speeds. However, the research that deals with such models, preemptive as well as nonpreemptive, most often has focused on approximations of the expected waiting times without taking any priorities into consideration: see, for example, Brandwajn and Begin [3]. Queueing models with priorities have received a fair amount of coverage in the literature. However, most priority queueing models have been subject to Poisson arrivals; see, for example, Kleinrock [7]. The most common objective function studied in priority queueing is the so-called expected weighted flow time. This has led to the well-known $c\mu$ rule in queueing theory; see, for example, Wolff [18]. Some research has been done on priority queueing models with due date related objective functions. Lehoczky [9] and Doytchinov et al. [5] have analyzed the performance of the Earliest Deadline First (EDF) rule in single server queues with due date related objective functions. Towsley and Panwar [17] and Liu and Towsley [12] have analyzed variations of the Earliest Deadline First rule in parallel server queues. They consider a preemptive environment with identical exponential servers in parallel with the due dates either known or stochastically comparable. Their objectives are the number of jobs among the first n jobs that have missed their deadline as well as the vector of tardinesses of the first n jobs. They show, taking advantage of the fact that their servers are identical and using sample path approaches (see Liu et al. [11]), that the preemptive EDD first policy minimizes their objectives stochastically. The main difference between our results and the Towsley and Panwar [17] and Liu and Towsley [12] framework is that our servers operate at different speeds. Having servers that operate at different speeds first requires more elaborate policies that have to take the due dates of the jobs as well as the speeds of the different machines into account. Having servers that operate at different speeds with multiple preemptions allowed also requires different proof techniques, since sample path approaches with coupling are now much harder to implement.

We will proceed as follows. First, in section 2 we construct our framework, including def-

initions and notation. In section 3, we begin presenting our core results, starting in section 3.1 with the optimality of the preemptive *EDD-FS* rule for the maximal lateness objective function, assuming first that the release and due dates are deterministic. We present a full proof of this result. Then, in sections 3.2 and 3.3 we present optimality results for the preemptive *EDD-FS* and *EDD-FS-P* rules for the sum of tardinesses and sum of unit penalties objective functions, respectively. Because the proofs of these two results are quite similar to that of the maximal lateness, we omit the full proofs and only discuss the key differences from the maximal lateness proof. In section 3.4, we generalize these results from deterministic to stochastic release and due dates. In section 4, we show the optimality of the *EDD-FS* and *EDD-FS-P* rules in a priority queueing setting for the long-term expected average tardiness and long-term expected proportion of jobs completed after their due dates. Finally, in section 5 we explore potential future work and generalizations of our results.

2. Notation and Preliminaries

The stochastic process of a server processing its jobs can be visualized as follows. Server i may be regarded as a Poisson process that generates events with a rate μ_i. If an event occurs while the server is processing a job, then the event corresponds to the completion of that job on the server. If an event generated by the Poisson process occurs at a time when the server is idle, then the generated event does not correspond to any event in the scheduling process. So server i is associated with the Poisson process N_i and if a job is being processed on server i at a jump time of N_i, then the job will be completed. Now let $N_1, \ldots, N_m$ be m independent Poisson processes with rates $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_m$ on a complete probability space $(\Omega, \mathcal{F}, P)$ and let $(\mathcal{F}_t, t \geq 0)$ be the completed natural filtration of these m Poisson processes.

Let

$$N(t) = \sum_{j=1}^{m} N_j(t) \tag{1}$$

and let $\tau_1, \tau_2, \ldots$ be the successive jump times of N.

Let $r_1 \leq \cdots \leq r_n$ be the release dates and $d_1, d_2, \ldots, d_n$ with $d_j > 0$, $j = 1, \ldots, n$, the due dates of the n jobs. The assumption that the due dates are positive do not constrain the level of generality, since the selection of the zero time point is arbitrary. So job j has release date r_j and due date d_j. The jobs are sorted by their release dates. We assume (without any loss of generality) that $r_1 > 0$. At time $t \geq 0$ we shall call job j *active* if $r_j \leq t$ and it has not been completed yet by time t (if

it has been completed exactly at time t then it is not considered active at t). We refer to job j at time t as a *future* job if $r_j > t$.

Next we introduce the class of so-called *admissible* policies. According to an admissible policy a scheduler will be allowed to assign selected available jobs to selected servers at stopping times (with respect to the filtration $(\mathcal{F}_t, t \geq 0)$). A policy π must include a sequence of stopping times $(A_l, l \geq 1)$ which will be referred to as *action times*. Action times are are the times when the manager interferes with the process, that is, takes some jobs off the machines on which they were running, assigns some jobs which were previously not running to some machines, assigns some sobs that are running on some machines to different machines, etc. Such action times satisfy the following properties:

(i) $0 \leq A_1 < A_2 < \cdots$;

(ii) $r_j \geq A_1 \Rightarrow r_j \in \{A_1, A_2, \ldots\}$ for all $j = 1, \ldots, n$;

(iii) $\tau_i \geq A_1 \Rightarrow \tau_i \in \{A_1, A_2, \ldots\}$ for all $i \geq 1$;

(iv) in any finite time interval the number of action times is finite, i.e., ($\lim_{k\to\infty} A_k = \infty$, a.s.)

The first property does not need any explanation. Properties (ii) and (iii) imply that, starting at A_1, each release date and each jump time of N must be an action time. We can require this without any loss of generality, since if any of these times were not included, we may include it, but not change the arrangement of the jobs on the servers at the newly added time. In particular, if any of the jump times happen before r_1 then obviously the policy does nothing. Extending the action times this way, we shall have exactly the same completion times as before.

At any action time A_l an admissible policy must specify:

(a) The number of jobs ν to be assigned to servers;

(b) A set of ν jobs with indexes $\alpha = (\alpha_1, \ldots, \alpha_\nu)$; the jobs are sorted so that $d_{\alpha(1)} \leq \cdots \leq d_{\alpha(\nu)}$.

(c) A set of ν servers $\beta = (\beta_1, \ldots, \beta_\nu)$ to which the set of ν jobs are assigned; the servers are sorted so that $\mu_{\beta(1)} \geq \cdots \geq \mu_{\beta(\nu)}$.

(d) A permutation $\gamma = (\gamma_1, \ldots, \gamma_\nu)$ of the set $\{1, \ldots, \nu\}$. The permutation determines the assignment of the ν jobs to the ν servers: job α_j is assigned to server $\beta_{\gamma(j)}$.

All these must be selected in an $\mathcal{F}_{A(l)}$-measurable fashion. So for any fixed combination of $\nu, \alpha, \beta, \gamma$, the event that this particular combination is selected at time A_l must be in the sigma field $\mathcal{F}_{A(l)}$. Since there are only finitely many possible combinations (ν cannot be larger than n), Ω must have a finite partition in $\mathcal{F}_{A(l)}$ such that for each event belonging to the partition, a fixed combination of $(\nu, \alpha, \beta, \gamma)$ is selected.

A policy π that satisfies all these requirements will be referred to as an *admissible* policy. Clearly, at any action time ν cannot be larger than the number of active jobs, and also $\nu \leq m$.

We say that a policy π is *locally EDD-FS* at time A_l on an event in $\mathcal{F}_{A(l)}$ if, at time A_l on the event the policy selects ν as the minimum of the number of active jobs and m, α as the index set of the ν jobs with the earliest due dates amongst the active jobs, β as the index set of the ν fastest servers, $\gamma = (1, 2, \ldots, \nu)$ as the permutation that assigns jobs to servers, i.e., $\gamma(j) = j$ and job $\alpha(j)$ is assigned to machine j.

We shall refer to the first action time A_1 corresponding to a policy π as the *initial time* for π. Let $\mathcal{V}$ be an arbitrary stopping time. We define the *EDD-FS* policy with initial time $\mathcal{V}$ by the requirements that the initial time for this policy A_1 is equal to $\mathcal{V}$, and that at every action time the policy is locally *EDD-FS* (on the entire Ω).

In the definition of the *EDD-FS* policy with initial time $\mathcal{V}$ we did not require that every action time must correspond to a release or a completion time of a job. However, by properties (ii) and (iii) of action times, all release dates and completion times are included in the set of action times. In principle, it is possible that there are additional action times (after $\mathcal{V}$), but at these additional times the *EDD-FS* policy will not change in any way the assignment of jobs to servers.

We now introduce notation that we will use throughout the paper. First, for a policy π, let $\bar{C}_1(\pi) < \cdots < \bar{C}_n(\pi)$ denote the successive completion times of the n jobs under π. Let $\bar{L}_j(\pi)$, $\bar{T}_j(\pi)$, and $\bar{U}_j(\pi)$ denote the lateness, tardiness, and unit penalty (respectively) of the job that has been completed at time $\bar{C}_j$, for $j \leq n$.

Second, for an admissible policy π, let the random variables $L(\pi)$, $T(\pi)$, and $U(\pi)$ denote the maximal lateness ($\max\{\bar{L}_1(\pi), \ldots, \bar{L}_n(\pi)\}$), the sum of the tardinesses ($\sum_{j=1}^n \bar{T}_j(\pi)$), and the sum of the unit penalties ($\sum_{j=1}^n \bar{U}_j(\pi)$), respectively.

Third, let F be a set of positive numbers, and $t \geq 0$ a fixed time. Suppose that we find ourselves at time t with a set of due dates for the active jobs given by F and future jobs given by the job index set $\{i : r_i > t\}$. We denote by $W(t, F)$ the maximal lateness for the active and future jobs at time t, assuming that at and after time t the policy uses the *EDD-FS* strategy. We allow here F to be the empty set as well.

Fourth, if $\mathcal{G}$ is a sigma-field, X is a random variable, and F is an event, we define $E[X|\mathcal{G}, F] = E[X1_F|\mathcal{G}]/P[F|\mathcal{G}]$ whenever the denominator on the right-hand side is positive, and let $E[X|\mathcal{G}, F] = 0$ whenever the denominator is zero.

Finally, we use the following notation for real numbers (or random variables) x and y,

$$x \vee y = \max(x, y) \quad \text{and} \quad x \wedge y = \min(x, y).$$

3. The Optimality of the *EDD-FS* and *EDD-FS-P* Rules

In subsections 3.1-3.3 we shall prove our results for the case of deterministic release dates and due dates. Then in Section 3.4 we shall generalize these results to stochastic release and due dates.

3.1. Maximal Lateness Objective Function

We now begin our presentation of results, starting with the maximal lateness objective function. Our first result is that the preemptive *EDD-FS* rule is the optimal policy with respect to the maximal lateness objective function. We present a full proof here.

Theorem 1. *The preemptive EDD-FS rule minimizes the objective $E(L_{\max})$. That is, for every admissible policy π,*

$$E[L_{\max}(\pi)] \geq E[L_{\max}(\textit{EDD-FS})], \tag{2}$$

where the EDD-FS policy has the same initial action time as π.

Proof. In order to carry out a proof based on induction, we shall prove a slightly stronger statement:

$$E[Y \vee L_{\max}(\pi)] \geq E[Y \vee L_{\max}(\textit{EDD-FS})], \tag{3}$$

for any $\mathcal{F}_{A(1)}$ measurable random variable Y such that $E[|Y|] < \infty$. This additional random variable is required for the induction step from $n-1$ to n jobs.

Clearly (3) implies (2), because for any admissible policy π we have

$$L_{\max}(\pi) \geq -\max\{d_j, j \leq n\}, \tag{4}$$

so by selecting $Y = -\max\{d_j, j \leq n\}$ in (3) we get (2).

We shall proceed by induction based on n.

For the case of $n = 1$ see Lemma 1 (in appendix).

Suppose now that (3) is true whenever there are only $n-1$ jobs. We need to show that it is true for n jobs as well. Let π be an arbitrary admissible policy for $n \geq 2$ jobs. We define policy π^0 as follows. Policy π^0 has the same action times (A_l) as π, does the same as π before time $\bar{C}_1(\pi)$, but at time $\bar{C}_1(\pi)$ and afterwards it behaves locally *EDD-FS* at every action time. From time $\bar{C}_1$ there are only $n-1$ jobs remaining, so we have

$$E\left[Y \vee L_{\max}(\pi)\right] = E\left[\max\{Y, \bar{L}_1(\pi), \bar{L}_2(\pi), \ldots, \bar{L}_n(\pi)\}\right] \geq$$

$$E\left[\max\{Y \vee \bar{L}_1(\pi), \bar{L}_2(\pi^0), \ldots, \bar{L}_n(\pi^0)\}\right] = E\left[Y \vee L_{\max}(\pi^0)\right].$$

In the above chain the inequality follows by applying our assumption for $n-1$ jobs, where the first action time is $\bar{C}_1$, and Y is replaced by $Y \vee L_1(\pi)$. Thus we have

$$E\left[Y \vee L_{\max}(\pi)\right] \geq E\left[Y \vee L_{max}\left(\pi^0\right)\right]. \tag{5}$$

We shall define a sequence of policies $(\pi^p, p = 0, 1, 2, \ldots)$ recursively in the following way. For $p \geq 1$, π^p has the same stopping times as π^{p-1} and behaves the same way as π^{p-1} before time A_p. At time A_p, it is locally *EDD-FS*. Also, π^p preserves the property that it is *EDD-FS* from the time of the first completion. Then

$$L_{\max}(\pi^p) \to L_{\max}(\textit{EDD-FS}) \text{ as } p \to \infty,$$

since $L_{\max}(\pi^p) = L_{\max}(\textit{EDD-FS})$ whenever p is larger than the number of action events before $\bar{C}_1(\pi^p)$, which is finite by property (iv). Suppose that we can show that for $p > 0$

$$E\left[Y \vee L_{\max}(\pi^p)\right] \leq E\left[Y \vee L_{\max}\left(\pi^{p-1}\right)\right], \tag{6}$$

Then, Fatou's lemma implies that

$$E\left[Y \vee L_{\max}(\textit{EDD-FS})\right] \leq E\left[Y \vee L_{\max}\left(\pi^0\right)\right], \tag{7}$$

and then (7) and (5) imply (3). Hence, in the rest of this proof we shall show that π^p can indeed be defined satisfying (6). We shall proceed by modifying π^{p-1} in four steps, creating π_1^p, π_2^p, π_3^p, and $\pi_4^p = \pi^p$, such that

$$E\left[Y \vee L_{\max}(\pi_i^p)\right] \leq E\left[Y \vee L_{\max}\left(\pi_{i-1}^p\right)\right], \tag{8}$$

for $i = 2, 3, 4$, and

$$E\left[Y \vee L_{\max}(\pi_1^p)\right] \leq E\left[Y \vee L_{\max}\left(\pi^{p-1}\right)\right]. \tag{9}$$

Here, we outline each step's purpose.

Step 1: selected jobs go to the fastest servers

Step 2: the number of servers used is the minimum of m and the number of active jobs (i.e., ν is the maximal possible value)

Step 3: selects jobs from the active jobs with the earliest due dates

Step 4: jobs with earlier due dates go to a faster server

Step 1. In this step we shall modify π^{p-1} so that at time A_p the selected jobs go to the fastest servers, that is, at A_p we always have $\beta = (1, 2, \ldots, \nu)$. Fix $0 \leq k \leq n$, and $\nu \leq k \wedge m$, $\alpha = (\alpha_1, \ldots, \alpha_\nu)$, $\beta = (\beta_1, \ldots, \beta_\nu)$, $\gamma = (\gamma_1, \ldots, \gamma_\nu)$ as described above. Let $i, j \leq \nu$. Let $H_1 = H_1(k, \nu, \alpha, \beta, \gamma, i, j)$ be the event on which the following are true:

i. $r_k \leq A_p < r_{k+1}$, where we define $r_0 = 0$ and $r_{n+1} = \infty$;

ii. At time A_p policy π^{p-1} selects $(\nu, \alpha, \beta, \gamma)$;

iii. There exists a job α_j for some $j \leq \nu$ and server $i \leq m$ such that job α_j goes to server $\beta(\gamma_j)$, $i \notin \{\beta_1, \ldots, \beta_\nu\}$, and $\mu_i \geq \mu_{\beta(\gamma(j))}$.

All jobs that arrived before or at A_p must be also active on H_1, because by condition 3 policy π^{p-1} is not locally *EDD-FS* at A_p on H_1, but we know that at and after the first completion π^{p-1} follows the *EDD-FS* rules. So none of the jobs $1, 2, \ldots, k$ could have completed by time A_p on H_1.

Suppose that $P(H_1) > 0$. Let $\tilde{\tau}_1$ be the time of the first jump of $N_{\beta(\gamma(j))}$ after A_p, and $\tilde{\tau}_2$ be the time of the first jump of N_i after A_p. We define $\tilde{\pi}$ in the following way. Policy $\tilde{\pi}$ has the same stopping times $A_1, A_2, \ldots$ as π^{p-1}, and it acts the same way as π^{p-1} at $A_1, \ldots, A_{p-1}$ (that is, locally *EDD-FS*). At time A_p if H_1 happens, then $\tilde{\pi}$ puts job α_j on server i, but for all $h \neq j$ it puts job α_h on server $\beta_{\gamma(h)}$, just like π^{p-1}. If H_1 does not happen at time A_p then $\tilde{\pi}$ behaves the same way as π^{p-1}. For $p' > p$ on $(H_1 \cap \{\tilde{\tau}_1 \wedge \tilde{\tau}_2 = A_{p+1}\})^c$ policy $\tilde{\pi}$ behaves the same way as π^{p-1}. On $H_1 \cap \{A_{p+1} = \tilde{\tau}_1\}$ policy $\tilde{\pi}$ behaves the same way at and after time A_{p+1} as π^{p-1} behaves at and after A_{p+1} on $H_1 \cap \{A_{p+1} = \tilde{\tau}_2\}$. On $H_1 \cap \{A_{p+1} = \tilde{\tau}_2\}$ policy $\tilde{\pi}$ behaves the same way at and after time A_{p+1} as π^{p-1} behaves at and after A_{p+1} on $H_1 \cap \{A_{p+1} = \tilde{\tau}_1\}$ (that is, *EDD-FS*). Essentially, $\tilde{\pi}$ is a pairwise interchange away from π^{p-1}.

Then on the event $H_1 \cap \{A(p+1) = \tilde{\tau}_1\}$ we have

$$L_{\max}(\pi^{p-1}) = \max\left\{A_{p+1} - d_{\alpha(j)}, W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_{\alpha(j)}\}\right)\right\}.$$

Let V be the first jump time of N after A_p. By Lemma 3 it follows that

$$E\left[\max\left\{Y, L_{\max}(\pi^{p-1})\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1 \cap \{A(p+1) = \tilde{\tau}_1 \wedge \tilde{\tau}_2\}\right] =$$

$$E\left[\max\left\{Y, A_{p+1} - d_{\alpha(j)}, W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_{\alpha(j)}\}\right)\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1, \{A_{p+1} = \tilde{\tau}_1\}\right] \times$$
$$\frac{\mu_{\beta(\gamma(j))}}{\mu_i + \mu_{\beta(\gamma(j))}} 1_{\{A(p+1)=V\}} 1_{H(1)} +$$
$$E\left[\max\left\{Y, L(\pi^{p-1})\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1, \{A_{p+1} = \tilde{\tau}_2\}\right] \frac{\mu_i}{\mu_i + \mu_{\beta(\gamma(j))}} 1_{\{A(p+1)=V\}} 1_{H(1)}. \quad (10)$$

However, given Y and A_{p+1}, the random variable

$$\max\left\{Y, A_{p+1} - d_{\alpha(j)}, W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_{\alpha(j)}\}\right)\right\}$$

is conditionally independent of $\mathcal{F}_{A(p+1)}$. Hence, we can write this identity as

$$E\left[\max\left\{Y, L_{\max}(\pi^{p-1})\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1 \cap \{A(p+1) = \tilde{\tau}_1 \wedge \tilde{\tau}_2\}\right] =$$

$$E\left[\max\left\{Y, A_{p+1} - d_{\alpha(j)}, W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_{\alpha(j)}\}\right)\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1, \{A_{p+1} = \tilde{\tau}_2\}\right] \times$$
$$\frac{\mu_{\beta(\gamma(j))}}{\mu_i + \mu_{\beta(\gamma(j))}} 1_{\{A(p+1)=V\}} 1_{H(1)} +$$
$$E\left[\max\left\{Y, L(\pi^{p-1})\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1, \{A_{p+1} = \tilde{\tau}_2\}\right] \frac{\mu_i}{\mu_i + \mu_{\beta(\gamma(j))}} 1_{\{A(p+1)=V\}} 1_{H(1)}. \quad (11)$$

We rewrite it this way to facilitate comparison with the corresponding expression for the $\tilde{\pi}$ policy below. Exactly the same way we can derive that

$$E\left[\max\left\{Y, L_{\max}(\tilde{\pi})\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1 \cap \{A(p+1) = \tilde{\tau}_1 \wedge \tilde{\tau}_2\}\right] =$$

$$E\left[\max\left\{Y, A_{p+1} - d_{\alpha(j)}, W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_{\alpha(j)}\}\right)\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1, \{A_{p+1} = \tilde{\tau}_2\}\right] \times$$
$$\frac{\mu_i}{\mu_i + \mu_{\beta(\gamma(j))}} 1_{\{A(p+1)=V\}} 1_{H(1)} +$$
$$E\left[\max\left\{Y, L(\tilde{\pi})\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1, \{A_{p+1} = \tilde{\tau}_1\}\right] \frac{\mu_{\beta(\gamma(j))}}{\mu_i + \mu_{\beta(\gamma(j))}} 1_{\{A(p+1)=V\}} 1_{H(1)}. \quad (12)$$

By the definition of $\tilde{\pi}$, we can write this last equation as

$$E\left[\max\left\{Y, L_{\max}(\tilde{\pi})\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1 \cap \{A(p+1) = \tilde{\tau}_1 \wedge \tilde{\tau}_2\}\right] =$$

$$E\left[\max\left\{Y, A_{p+1} - d_{\alpha(j)}, W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_{\alpha(j)}\}\right)\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1, \{A_{p+1} = \tilde{\tau}_2\}\right] \times$$

$$\frac{\mu_i}{\mu_i + \mu_{\beta(\gamma(j))}} 1_{\{A(p+1)=V\}} 1_{H(1)} +$$

$$E\left[\max\left\{Y, L(\pi^{p-1})\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1, \{A_{p+1} = \tilde{\tau}_2\}\right] \frac{\mu_{\beta(\gamma(j))}}{\mu_i + \mu_{\beta(\gamma(j))}} 1_{\{A(p+1)=V\}} 1_{H(1)}. \tag{13}$$

Then, by the induction hypothesis,

$$E\left[\max\left\{Y, A_{p+1} - d_{\alpha(j)}, W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_{\alpha(j)}\}\right)\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1, \{A_{p+1} = \tilde{\tau}_2\}\right] \leq$$

$$E\left[\max\left\{Y, L(\pi^{p-1})\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1, \{A_{p+1} = \tilde{\tau}_2\}\right] \tag{14}$$

Indeed, by the induction hypothesis, the left-hand side would be dominated by the right-hand side if on the right-hand side we disregarded job α_j, because there would be only $n-1$ jobs remaining. Including job α_j would not decrease the right-hand side. Thus, $\mu_i > \mu_{\beta(\gamma(j))}$, (11), (13), and (14) imply

$$E\left[\max\left\{Y, L_{\max}(\pi^{p-1})\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1 \cap \{A(p+1) = \tilde{\tau}_1 \wedge \tilde{\tau}_2\}\right] \geq$$

$$E\left[\max\left\{Y, L_{\max}(\tilde{\pi})\right\} \mid \mathcal{F}_{A(p)}, A_{p+1}, H_1 \cap \{A(p+1) = \tilde{\tau}_1 \wedge \tilde{\tau}_2\}\right].$$

Since $L_{\max}(\pi^{p-1})$ and $L_{\max}(\tilde{\pi})$ coincide on $(H_1 \cap \{A(p+1) = \tilde{\tau}_1 \wedge \tilde{\tau}_2\})^c$, thus

$$E\left[Y \vee L_{\max}(\pi^{p-1})\right] \geq E\left[Y \vee L_{\max}(\tilde{\pi})\right].$$

Repeating this procedure, we reach a policy for which $\beta = \{1, 2, \ldots, \nu\}$ on H_1. Repeating this for every combination of $k, \nu, \alpha, \beta, \gamma$, we reach policy π_1^p, which puts all selected jobs on the fastest servers at time A_p, and (9) holds

Step 2. In this step, we modify π_1^p so that ν is the maximal possible value at time A_p, that is, either equal to the number of active jobs, or to m, whichever is smaller. For fixed $k \leq n$, $\nu < k \wedge m$, $\alpha = (\alpha_1, \ldots, \alpha_\nu)$, and $\gamma = (\gamma_1, \ldots, \gamma_\nu)$, we define the event H_2 such that on H_2 item 1 from the definition of H_1 holds, but we replace item 2 by

2*: At time A_p policy π_1^p selects (ν, α, γ).

(There is no third item in this definition).

Notice that we assumed $v < k \wedge m$, so π_1^p does not put up the maximal possible number of jobs on the event H_1. Also, it is not following the *EDD-FS* policy in that case, so one can argue similarly to the corresponding part in step 1 that all k jobs that arrived by that time are active.

In item 2*, we left out β, since at A_p policy π_1^p already selects the fastest servers. Since

$\nu < k \wedge m$, there exists a job, say job i, such that $i \notin \{\alpha_1, \ldots, \alpha_\nu\}$. We define policy $\hat{\pi}$ in the following way. It has the same stopping times as π_1^p, and behaves the same way as π_1^p at times $A_1, \ldots, A_{p-1}$. At time A_p if H_2 happens then policy $\hat{\pi}$ selects job i in addition to $\{\alpha_1, \ldots, \alpha_\nu\}$ and puts these jobs on the fastest $\nu + 1$ servers, using a permutation $\hat{\gamma}$ such that job α_u goes on server γ_u for $u \leq \nu$ and job i goes on server $\nu + 1$. On H_2^c policy $\hat{\pi}$ behaves the same way at time A_p as π_1^p. Let B_1 be the event that at time A_{p+1} the process $N_{\nu+1}$ has a jump. For $p' > p$ on $(H_2 \cap B_1)^c$ policy $\hat{\pi}$ behaves the same way as π_1^p. Let $\bar{C}_1(\pi_1^p)$ be the first completion of a job after A_p under π_1^p (which is actually its overall first completion). On $H_2 \cap B_1$ at any action time $A_{p'}$ such that $A_{p'} < \bar{C}_1(\pi_1^p)$ and $p' > p$, policy $\hat{\pi}$ behaves just like π_1^p, except whenever π_1^p puts job i on a server, policy $\hat{\pi}$ leaves that server empty. At and after $\bar{C}_1(\pi_1^p)$ policy $\hat{\pi}$ is locally *EDD-FS*. We have $L_{\max}(\pi_1^p) = L_{\max}(\hat{\pi})$ on $(H_2 \cap B_1)^c$. Let $\bar{l}$ be the number of jobs releasing between times A_p and $\bar{C}_1(\pi_1^p)$. Let $\tilde{B}_u$ be the event that under π_1^p it is job u that is completed at time $\bar{C}_1(\pi_1^p)$.Then on the event $H_2 \cap B_1 \cap \tilde{B}_i$

$$L_{\max}(\pi_1^p) = \max\left\{\bar{C}_1(\pi_1^p) - d_i, W\left(\bar{C}_1(\pi_1^p), \{d_1, \ldots, d_{k+\bar{l}}\} \setminus \{d_i\}\right)\right\}$$

and

$$L_{\max}(\hat{\pi}) = \max\left\{A_{p+1} - d_i, W\left(\bar{C}_1(\pi_1^p), \{d_1, \ldots, d_{k+\bar{l}}\} \setminus \{d_i\}\right)\right\}$$

hence by $A_{p+1} < \bar{C}_1(\pi_1^p)$ we have $L(\pi_1^p) \geq L_{\max}(\hat{\pi})$ in this case. On the event $H_2 \cap B_1 \cap \tilde{B}_u$, for $u \neq i$

$$L_{\max}(\pi_1^p) = \max\left\{\bar{C}_1(\pi_1^p) - d_u, W\left(\bar{C}_1(\pi_1^p), \{d_1, \ldots, d_{k+\bar{l}}\} \setminus \{d_u\}\right)\right\}$$

and

$$L_{\max}(\hat{\pi}) = \max\left\{A_{p+1} - d_i, \bar{C}_1(\pi_1^p) - d_u, W\left(\bar{C}_1(\pi_1^p), \{d_1, \ldots, d_{k+\bar{l}}\} \setminus \{d_u, d_i\}\right)\right\}$$

From $A_{p+1} < \bar{C}_1(\pi_1^p)$ it follows that

$$A_{p+1} - d_i \leq W\left(\bar{C}_1(\pi_1^p), \{d_1, \ldots, d_{k+\bar{l}}\} \setminus \{d_u\}\right).$$

From Lemma 5 it follows that

$$W\left(\bar{C}_1(\pi_1^p), \{d_1, \ldots, d_{k+\bar{l}}\} \setminus \{d_u, d_i\}\right) \leq W\left(\bar{C}_1(\pi_1^p), \{d_1, \ldots, d_{k+\bar{l}}\} \setminus \{d_u\}\right)$$

hence $L_{\max}(\hat{\pi}) \leq L_{\max}(\pi_1^p)$, a.s. We repeat this step until we reach a policy such that $\nu = k \wedge m$ on H_2. Then, repeating this for every combination of k, ν, α, γ we reach policy π_2^p which satisfies

the requirements.

Step 3. In this step we define π_3^p by modifying π_2^p so that at time A_p it selects the jobs from the active jobs with the earliest due dates. For fixed $k \leq n$, $\alpha = (\alpha_1, \ldots, \alpha_{m \wedge k})$, $\gamma = (\gamma_1, \ldots, \gamma_{m \wedge k})$, $i \leq k$, $j \leq m$ we define the event H_3 such that on H_3 item 1 from the definition of H_1 holds, but we replace item 2 and 3 by

2**: At time A_p policy π_2^p selects (α, γ).

3**: $m < k$ and there exist an active job $i \leq k$ such that $i \notin (\alpha_1, \ldots, \alpha_m)$, and a job $\alpha_j \in \alpha$ such that $d(\alpha_1) \leq \ldots \leq d(\alpha_{j-1}) \leq d(i) < d(\alpha_j) \leq \ldots \leq d(\alpha_m)$.

We define policy $\tilde{\pi}$ in the following way. It has the same stopping times as π_2^p, and behaves the same way at $A_1, \ldots, A_{p-1}$. At time A_p if H_3 happened then $\tilde{\pi}$ selects jobs $\{\alpha_1, \ldots, \alpha_{j-1}, i_s, \alpha_{j+1}, \ldots, \alpha_m\}$, and uses the same permutation γ that π_2^p uses. On H_3^c policy $\tilde{\pi}$ at time A_p behaves the same way as π_2^p. Let B_2 be the event that it is $N_{\gamma(j)}$ that jumps at time A_{p+1}. For $p' > p$ on $(H_3 \cap B_2)^c$ policy $\tilde{\pi}$ behaves the same way at time $A_{p'}$ as π_2^p. On $H_3 \cap B_2$ policy $\tilde{\pi}$ is locally *EDD-FS* at $A_{p'}$ for $p' > p$. Then on $(H_3 \cap B_2)^c$ we have $L_{\max}(\tilde{\pi}) = L_{\max}(\pi_2^p)$. On $H_3 \cap B_2$

$$L_{\max}(\pi_2^p) = \max\left\{A_{p+1} - d_{\alpha(j)}, W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_{\alpha(j)}\}\right)\right\}$$

and

$$L_{\max}(\tilde{\pi}) = \max\left\{A_{p+1} - d_i, W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_i\}\right)\right\}.$$

Clearly we have

$$A_{p+1} - d_i \leq W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_{\alpha(j)}\}\right)$$

and, by Lemma 4,

$$W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_i\}\right) \leq W\left(A_{p+1}, \{d_1, \ldots, d_k\} \setminus \{d_{\alpha(j)}\}\right)$$

thus $L_{\max}(\tilde{\pi}) \leq L_{\max}(\pi_2^p)$, a.s. We repeat this step until we reach a policy that on H_3 selects the m jobs out of the k active jobs with the earliest due dates. Then, repeating this for every combination of k, α, γ, i, j we reach policy π_3^p which satisfies the requirements.

Step 4. In this step we define $\pi_4^p = \pi^p$ by modifying π_3^p so that at time A_p the job with the earlier due date goes to a faster server. For fixed $k \leq n$, and $\gamma = (\gamma_1, \ldots, \gamma_{m \wedge k})$ we define the event H_4 such that on H_4 item 1 from the definition of H_1 holds, but we replace item 2 by

2***: At time A_p policy π_3^p selects permutation γ.

(There is no third condition for H_4.)

Let $(i_1, \ldots, i_k)$ be a permutation of the k active jobs such that

$$d_{i(1)} \leq \ldots \leq d_{i(k)}. \tag{15}$$

On H_4 policy π_3^p selects jobs $\{i_1, \ldots, i_{m\wedge k}\}$, and job i_j goes to server γ_j for $j \leq k \wedge m$. Let V be the first jump time of N after A_p, and let $\bar{B}_s$ be the event that N_s jumps at time V. Let $\bar{B} = \cup_{s\leq m\wedge k}\bar{B}_s$. We define π^p in the following way. It has the same stopping times as π_3^p, and acts the same way as π_3^p at $A_1, \ldots, A_{p-1}$. On the event $(H_4 \cap \{A_{p+1} = V\} \cap \bar{B})^c$ policy π^p behaves the same way as π_3^p at time A_p and afterwards. On the event $H_4 \cap \{A_{p+1} = V\} \cap \bar{B}$ policy π^p selects the permutation $\bar{\gamma} = (1, 2, \ldots, k \wedge m)$, and after A_p is locally *EDD-FS*. From this construction, it follows that on $(H_4 \cap \{A_{p+1} = V\} \cap \bar{B})^c$ we have $L(\pi^p) = L(\pi_3^p)$. On the event $H_4 \cap \{A_{p+1} = V\} \cap \bar{B}_s$ we have for $s \leq k \wedge m$

$$L_{\max}(\pi_3^p) = \max\left\{V - d_{i(\gamma^{-1}(s))}, W\left(V, \mathcal{D} \setminus \{d_{i(\gamma^{-1}(s))}\}\right)\right\},$$

where $\mathcal{D} = \{d_1, \ldots, d_k\}$. Then by Lemma 6

$$E[\max\{Y, L(\pi_3^p)\}|\mathcal{F}_V]1_{\{A(p+1)=V\}\cap H_4\cap\bar{B}} =$$

$$\sum_{s\leq m\wedge k} E\left[\max\left\{Y, V - d_{i(\gamma^{-1}(s))}, W\left(V, \mathcal{D} \setminus \{d_{i(\gamma^{-1}(s))}\}\right)\right\}|\mathcal{F}_V\right] 1_{\{A(p+1)=V\}\cap H_4\cap\bar{B}_s} =$$

$$= \sum_{s\leq m\wedge k} F_{\gamma^{-1}(s)}(Y, V)1_{\{A(p+1)=V\}\cap H_4\cap\bar{B}_s},$$

where for $y \in \mathbb{R}$ and $v \geq 0$

$$F_j(y, v) = E\left[\max\left\{y, v - d_{i(j)}, v + W\left(V, \mathcal{D} \setminus \{d_{i(j)}\}\right) - V\right\}\right]. \tag{16}$$

By Lemma 2

$$E[\max\{Y, L_{\max}(\pi_3^p)\}1_{\{A(p+1)=V\}\cap H_4\cap\bar{B}}|\mathcal{F}_{A(p)}, V] =$$

$$\sum_{s\leq m\wedge k} F_{\gamma^{-1}(s)}(Y, V)1_{\{A(p+1)=V\}\cap H_4}\frac{\mu_s}{\sum_{r\leq m}\mu_r}.$$

One can derive a similar formula exactly the same way for π^p, except that $\gamma^{-1}(s)$ must be replaced by s. Since on $(H_4 \cap \{A_{p+1} = V\} \cap \bar{B})^c$ we have $L(\pi_3^p) = L(\pi^p)$, in order to conclude that $E[\max\{Y, L(\pi_3^p)\}|\mathcal{F}_{A(p)}, V] \geq E[\max\{Y, L(\pi^p)\}|\mathcal{F}_{A(p)}, V]$ all we need to show that

$$F_j(y, v) \leq F_{j+1}(y, v) \text{ for } j < k \wedge m. \tag{17}$$

By Lemma 4 and (15)

$$W(V, \mathcal{D} \setminus \{d_{i(j)}\}) \leq W(V, \mathcal{D} \setminus \{d_{i(j+1)}\})$$

and clearly

$$v - d_{i(j)} \leq v + W(V, \mathcal{D} \setminus \{d_{i(j+1)}\}) - V$$

so (17) follows. □

3.2. Sum of Tardinesses Objective Function

Our second result is that the preemptive *EDD-FS* rule is again the optimal policy with respect to the sum of tardinesses objective function.

Theorem 2. *The preemptive EDD-FS rule minimizes the objective* $E(\sum T_j)$. *That is, for every admissible policy* π,

$$E\left[T(\pi)\right] \geq E\left[T\left(\textit{EDD-FS}\right)\right], \tag{18}$$

where the EDD-FS policy has the same initial action time as π.

Our proof for the sum of tardinesses objective function is quite similar to that of the maximal lateness. So, we omit the full proof for brevity and only summarize the key differences.

Naturally, the most important difference is that the maximal lateness expression must be replaced with the sum of tardinesses expression. Second, this objective function does not require a structure like (3). The auxiliary random variable Y can be dropped. Third, the lemmas require small adjustments.

Apart from these differences, the structure and flow of the proofs are nearly identical.

3.3. Sum of Unit Penalties Objective Function

The *EDD-FS* rule with the addition of postponements (*EDD-FS-P*), is also the optimal policy with respect to the sum of unit penalties objective function.

Theorem 3. *The preemptive EDD-FS-P rule minimizes* $E(\sum U_j)$. *That is, for every admissible policy* π,

$$E\left[U(\pi)\right] \geq E\left[U\left(\textit{EDD-FS-P}\right)\right], \tag{19}$$

where the EDD-FS-P policy has the same initial action time as π.

Our proof for this objective function is quite similar in structure and flow to that of the sum of tardinesses and the maximal lateness objective functions. This requires amendments to the preliminaries used for those two proofs.

The first amendment relates to the notation of the completion times C. Here, let C_j be the minimum of job j's completion time and its due date. In other words, if job j does not get completed by time d_j, then $C_j = d_j$, at which point that job is discarded. If job j is completed before time d_j, then C_j is equal to the time of its completion. We shall call C_j the completion/discarding time of job j. The unit penalty of a job is the indicator of the job not completing by its due date. So, the unit penalty of job j is

$$U_j = \begin{cases} 1 \text{ if } C_j = d_j \\ 0 \text{ otherwise} \end{cases} .$$

The second amendment is an addition to the requirements of action times. To property (ii), we add a condition that requires all due dates to be action times. So, property (ii) is now:

(ii) For all $j = 1, \ldots, n$ we have $r_j \geq A_1 \Rightarrow r_j \in \{A_1, A_2, \ldots\}$ and $d_j \geq A_1 \Rightarrow d_j \in \{A_1, A_2, \ldots\}$. Properties (i), (iii), and (iv) are unchanged.

The third and final amendment is the addition of a new category of jobs: discarded jobs. A job is called *discarded* at time t if $d_j \leq t$. Additionally, the definition of active jobs changes to the following: a job is called *active* if $r_j \leq t$ and $d_j > t$.

Since completing a job after its due date results in a unit penalty value of 1 regardless of the amount of time between the completion time and the due date, we assume without loss of generality that policies never assign discarded jobs to servers.

Note that a family of policies that prioritizes incomplete jobs with future due dates over incomplete jobs with past due dates is equivalent to our admissible (discarding) policies with respect to the sum of unit penalties objective function. Such a priority policy would only assign the lower priority jobs to a server when that server would otherwise be idle. The objective function will be the same if the policy discards jobs with past due dates, or assigns it to zero priority.

Specifically, we would construct the priority system as follows. For all times $t > 0$ and for all jobs j, $j = 1, \ldots, n$, if job j is active and $d_j > t$, then job j's priority is 1. Otherwise, the job has priority 0. We would require that at any action time A_l, the policy satisfies $\gamma_j \geq \gamma_k$ whenever α_j is a priority 1 job, and α_k is a priority 0 job (i.e., a priority 0 job cannot be placed on a faster machine than a priority 1 job).

Priority policies would be more robust because they are still efficient policies and would minimize the expected makespan in addition to the sum of unit penalties. However, we choose to work with discarding policies to keep the proof simpler and more consistent with those of the maximal lateness and sum of tardinesses objective functions.

With these amendments, the proof for the sum of unit penalties objective function is similar to those of the maximal lateness and sum of tardinesses objective functions. So, we omit the proof for brevity.

3.4. Back to the Stochastic Release and Due Dates

In our proofs in the previous sections we assumed that the release dates r_j and the due dates d_j of the jobs are deterministic. However, we can easily generalize these results to the case when the release and due dates are random, assuming that the due date of job j is set and announced at the time when job j is released. However, we need to assume that the release and due dates are independent of the Poisson processes representing the servers. In this case we can condition on the release and due dates and use our previous results under the conditional probability measures. Our optimal policies (the *EDD-FS* and the *EDD-FS-P* policies) in the deterministic case do not require the knowledge of the release and due dates of future jobs, and we already know the optimality under the conditional probability measure given any outcome of those dates. Thus, the optimality in the random case follows.

4. Applications to Priority Queues

Since in this section the release date (arrival time) and due date of job j are considered to be random variables, we denote these here by R_j and D_j, respectively.

We shall examine the limiting situation when there are infinitely many jobs, and we assume that all policies are *efficient*. Efficiency means that (i) at every action time the number of jobs ν the policy puts up is the minimum of the number of active jobs at the time and m; (ii) at any action time the ν jobs selected to be put up on machines, are put on the ν fastest machines.

We shall consider the following queueing model with m parallel servers that operate at different speeds. The jobs arrive at the system according to an arbitrary renewal process with a mean interarrival time $1/\lambda$, and we assume that this arrival process is independent of the Poisson processes representing the servers. Our basic assumption is that the utilization factor of the queue is less than 1, i.e.,

$$\rho = \frac{\lambda}{\sum_{i=1}^{m} \mu_i} < 1. \tag{20}$$

Since the processing time distributions have finite moments, we know that the expected time the jobs remain in the system is finite and that any efficient policy minimizes this expected time.

We assume that the random due date D_j is set at the arrival of job j such that $(D_j - R_j,\ j \geq 1)$ forms an i.i.d. sequence of random variables, independent of the arrival times and of the Poisson processes. We shall call the time interval $D_j - R_j$ the *time window* of job j. At the arrival of a job its due date becomes known to the scheduler. However, its processing requirement is still unknown at that time, and only becomes known upon its completion.

Since we are now considering a queueing setting, we are interested in long-term averages of the performance measures, namely the mean tardiness of a job in steady state and the proportion of jobs (i.e., the percentage) that are completed after their due date. Formally, these are given by

$$\lim_{n\to\infty} E\left(\frac{\sum_{j=1}^{n} \bar{T}_j(\pi)}{n}\right), \tag{21}$$

and

$$\lim_{n\to\infty} E\left(\frac{\sum_{j=1}^{n} \bar{U}_j(\pi)}{n}\right), \tag{22}$$

where $\bar{T}_j$ and $\bar{U}_j$ are the tardiness and the unit penalty of the job completed at the j^{th} successive completion time $\bar{C}_j$. So the first performance measure is the long-term expected average tardiness of the jobs in the system and the second performance measure is the long-term expected proportion of jobs completed after their due dates. The first performance measure is clearly bounded by the limit of the average expected waiting time of the jobs in the system under any efficient policy whenever $D_j \geq R_j$ for all j. The second performance measure is a number between 0 and 1. Clearly both performance measures are heavily affected by the distribution function of the time windows.

We would like to show that the *EDD-FS* and *EDD-FS-P* policies on an infinite horizon minimize these two performance measures. However, one has to be careful here. It may be a bit surprising, but for fixed n the *EDD-FS-P* on an infinite horizon does not minimize the value of $E\left[\sum_{j=1}^{n} \bar{U}_j\right]$. In order to elucidate this fact we describe a particular situation. Suppose that $n-1$ jobs have already been completed, and there are 2 jobs in the queue, one with a due date very close, the other with a due date extremely far off. In this case the optimal policy for the above loss function would assign the job with the latest due date on the fastest machine and not assign the other job at all. This is very different from the actions of the *EDD-FS-P* policy. Similarly, for fixed n the *EDD-FS* policy on an infinite horizon does not minimize the value of $E\left[\sum_{j=1}^{n} \bar{T}_j\right]$. The situation is similar if we want to minimize $E\left[\sum_{j=1}^{n} T_j\right]$ or $E\left[\sum_{j=1}^{n} U_j\right]$ for fixed n, that is, the sum of the penalty functions for the first n jobs sorted by arrival. The infinite horizon *EDD-FS* and *EDD-FS-P* policies do not minimize these loss functions because these policies would put up

jobs that arrive after the arrival of the n^{th} job but before all those n jobs are completed. But such policy would be suboptimal for these loss functions with fixed n.

In order to remedy this situation, we define *cycles* $[R_{M(k)}, R_{M(k+1)})$ for $1 = M_1 < M_2 < \cdots$ by recursion. Let $M_1 = 1$, and let $R_{M(k+1)}$ be the first release date strictly after $R_{M(k)}$ such that just before $R_{M(k+1)}$ there are no active jobs in the system, but at time $R_{M(k+1)}$ there is exactly one active job in the system. Formally, if q_t is the right-continuous process representing the number of active jobs at time t (recall the definition of active jobs from the paragraph below formula (1), then

$$M_{k+1} = \inf\{k > M_k : q_{R(k)-} = 0 \text{ and } q_{R(k)} = 1\}.$$

Note that in the queueing literature such a cycle is typically referred to as a busy period. After a moment of reflection one may be convinced that the sequence of cycles is the same for every efficient policy. From (20) it follows that $M_k < \infty$ for every $k \geq 1$. Let π be any efficient policy. Let $T^k(\pi)$ be the total tardiness in the k^{th} cycle under policy π, i.e., the sum of tardinesses for jobs that arrive in the k^{th} cycle. Similarly, let $U^k(\pi)$ denote the number of late jobs in the k^{th} cycle under policy π, i.e., the number of jobs that arrive in the k^{th} cycle and are completed after their due dates. The following theorem states that the *EDD-FS* policy minimizes the long-term average tardiness within the cycles and that the *EDD-FS-P* policy minimizes the long-term proportion of the number of late jobs within the cycles.

Theorem 4. *Assume (20), and let π be an efficient policy. Then*

$$\liminf_n E\left[\frac{1}{n}\sum_{k=1}^{n} T^k(\pi)\right] \geq \lim_n E\left[\frac{1}{n}\sum_{k=1}^{n} T^k(\textit{EDD-FS})\right], \tag{23}$$

and

$$\liminf_n E\left[\frac{1}{n}\sum_{k=1}^{n} U^k(\pi)\right] \geq \lim_n E\left[\frac{1}{n}\sum_{k=1}^{n} U^k(\textit{EDD-FS-P})\right].$$

Proof. Notice that the sequence $(T^k(\textit{EDD-FS}), k \geq 1)$ is an i.i.d. sequence of random variables, so the right-hand side of (23) is equal to $E[T^1(\textit{EDD-FS})]$. On the other hand, one can show using a method quite similar to the finite case that $E[T^k(\pi)] \geq E[T^k(\textit{EDD-FS})] = E[T^1(\textit{EDD-FS})]$, so the first part of the theorem follows. The second identity can be shown similarly. □

Suppose now that, in addition to being efficient, policies are also *cycle-stationary*. By this we mean that the policy bases its decisions only on elapsed times of random events that have been observed in the current cycle since the beginning of the current cycle (these events are the jump times of each Poisson process and the release and due dates of jobs within the current cycle up to

the present time), and given this information the policy makes the same decisions in every cycle, i.e., the policy does not change from cycle to cycle. For cycle-stationary efficient policies one can show similarly to the above theorem that the limit of the expected average tardiness (see (21)) and the limit of the expected proportion of tardy jobs (see (22)) are minimized by the *EDD-FS* and the *EDD-FS-P* policies, respectively.

Note that we have not considered the $L_{\max}$ objective in this section. The reason for that is the following. Let $\bar{L}_j(\pi)$ be the lateness of the job that gets completed at time $\bar{C}_j(\pi)$. Then for all cycle-stationary policies

$$\lim_{n\to\infty} \max_{j\le n} \bar{L}_j(\pi) = \infty, \quad \text{a.s.}$$

5. Conclusions and Extensions

In the previous sections we have considered various stochastic scheduling problems and priority queueing models for which simple and mathematically elegant policies are optimal. This is in contrast to their deterministic counterparts for which no simple preemptive priority rules exist. It may be the case that there are more deterministic scheduling problems with equal processing requirements that are computationally very hard, but that have counterparts with exponentially distributed processing requirements and with elegant optimal policies. The i.i.d. exponentially distributed processing requirements are a crucial assumption, since the policies are not optimal when the processing requirements are i.i.d. with say a Decreasing Failure Rate (DFR) distribution. We have seen other examples of scheduling models where this is also the case, e.g., Pinedo and Reed [15]. It may be of interest in the future to:

(i) Determine scheduling conditions and objectives in a preemptive parallel machine environment under which nice mathematically elegant policies are optimal when the n jobs have random processing times that are i.i.d. exponentially distributed.

(ii) Determine (the less general) scheduling conditions and objectives in a preemptive parallel machine environment under which nice mathematically elegant policies are optimal when either the n jobs have equal deterministic processing times or the n jobs have random processing requirements that are i.i.d. exponentially distributed.

The following question may also be raised: are there any other processing time distributions, e.g., hyperexponential distributions, for which similar scheduling problems have optimal rules that can be determined relatively easily? The answer is affirmative for the following very special case of a hyperexponential distribution. Consider a processing time distribution of job j with the

processing time requirement being 0 (i.e., exponentially distributed with rate ∞) with probability q and exponentially distributed with mean 1 with probability $1-q$. One can easily determine the optimal policies for the models considered in this paper with this processing time distribution. In this case, at each job arrival the job is immediately put on a machine for a split second, independent of its due date. If the processing time turns out to be zero, then the job is completed and leaves the system. If it is not zero, then we know that the distribution of its remaining processing time is exponential with mean one. So each job in the system at any point in time has the same exponentially distributed remaining processing time requirement with mean 1. So our policies can then be applied to those jobs that have nonzero processing time requirements. However, even though it is easy to characterize the optimal policies in this case, these optimal policies are different from the *EDD-FS* and the *EDD-FS-P* policies described in the previous sections.

One may be able to generalize the results obtained in the previous sections in certain directions. One can consider, for example, the following generalization of the models considered. Assume job j has a weight w_j and a cost $w_j f_j(C_j)$ that is incurred upon completion of the job at time C_j. The function $f_j(C_j)$ is increasing in C_j. The objective to be minimized is $\sum w_j f_j(C_j)$. We say that function f_j of job j is *steeper* than function f_k of job k if

$$\frac{df_j(t)}{dt} \geq \frac{df_k(t)}{dt}$$

for all t. We use for this *steepness order* the notation $f_j \geq_s f_k$. One may show that if the weights w_j and the cost functions f_j are agreeable with one another in such a way that a higher job weight always goes with a steeper cost function, then the preemptive rule which at all times assigns the job with the *Largest Weight on the Fastest Machine (LW-FM)* minimizes the objective $E(\sum w_j f_j(C_j))$. If the weights of two jobs are equal, then the job with the steeper cost function has priority over the other job. It is clear that such a result would be a generalization of the result of the sum of the tardinesses presented in section 3.2. It is easy to see that the two tardiness cost functions T_j and T_k of jobs j and k are steepness ordered (the job with the earlier due date has a steeper cost function). The result would depend strongly on the agreeability condition. (Without the agreeability condition it is not clear what the optimal policy would be.) A formal proof of such a result would be somewhat cumbersome and would require a fair amount of additional notation. (A proof of such a result could be based on a technique described in Section 10.4 of Pinedo [16].)

However, note that a similar generalization cannot be established for the objective function $\sum w_j U_j$. Consider two cost functions f_j and f_k that are everywhere differentiable and very close

approximations of two unit penalty cost functions U_j and U_k with different due dates. It is clear that the functions f_j and f_k are not steepness ordered.

Appendix

Lemma 1. *In the case of $n = 1$ (there is only 1 job) (3) holds.*

This lemma claims that for $n = 1$ the optimal policy is to put the single job on the fastest server, and leave it there until completion. Of course intuitively this is quite obvious. The exact proof of this is quite similar to the proofs in steps 1 and 2, except that this case is much simpler. For brevity we omit the details.

Lemma 2. *Let A_p be an action time, and let V be the first jump of N after A_p. Then $\{A_{p+1} = V\} \in \sigma(\mathcal{F}_{A(p)}, V)$.*

Proof. Let Q be the set of rational numbers. Then

$$\{A_{p+1} < V\} = \cup_{n\geq 1} \cup_{q\in Q} \{\tau_n \leq A_p, A_{p+1} < q < \tau_{n+1}\} =$$

$$\cup_{n\geq 1} \cup_{q\in Q} \{\tau_n \leq A_p, A_{p+1} < q < \tau_{n+1}, A_p < q < V, A_p < \tau_{n+1}\}. \tag{24}$$

By Jacod and Shiryaev [6] Lemma III.1.29a, for every $n \in \mathbb{N}$ there exists an event $B_n \in \mathcal{F}_{\tau(n)}$ such that $\{A_{p+1} < q\} \cap \{q < \tau_{n+1}\} = B_n \cap \{q < \tau_{n+1}\}$. Then (24) can be written as

$$\cup_{n\geq 1} \cup_{q\in Q} B_n \cap \{\tau_n \leq A_p\} \cap \{q < \tau_{n+1}\} \cap \{A_p < \tau_{n+1}, A_p < q\} \cap \{q < V\}.$$

But $\{\tau_n \leq A_p < q < V\} \cap \{A_p < \tau_{n+1}\}$ implies $\{q < \tau_{n+1}\}$, hence we can write the above expression as

$$\cup_{n\geq 1} \cup_{q\in Q} B_n \cap \{\tau_n \leq A_p\} \cap \{A_p < \tau_{n+1}, A_p < q\} \cap \{q < V\}.$$

The events $B_n \cap \{\tau_n \leq A_p\}$ and $\{A_p < \tau_{n+1}, A_p < q\}$ are both members of $\mathcal{F}_{A(p)}$, hence the statement of the lemma follows. □

Lemma 3. *Let A_p be an action time, and let V be the first jump time of N after A_p. Let $i, j \leq m$ be two servers. Let $\tilde{\tau}_i$ be the first jump time of N_i after A_p, and let $\tilde{\tau}_j$ be the first jump time of N_j after A_p. Let H be an arbitrary event in $\mathcal{F}_{A(p)}$. Then*

$$P[A_{p+1} = \tilde{\tau}_i | \mathcal{F}_{A(p)}, A_{p+1}, H, \{A_{p+1} = \tilde{\tau}_i \wedge \tilde{\tau}_j\}] = \frac{\mu_i}{\mu_i + \mu_j} 1_{\{A_{p+1}=V\}} 1_H. \tag{25}$$

Proof. By the requirement that every jump of N is an action time and by Lemma 2 the left-hand

side of (25) can be written as

$$1_{\{A(p+1)=V\}} 1_H P(V = \tilde{\tau}_i | \mathcal{F}_{A(p)}, V, \{V = \tilde{\tau}_i \wedge \tilde{\tau}_j\}),$$

which is equal to the right-hand side of (25) by the elementary properties of the Poisson process. □

Lemma 4. *Let $k \geq 1$, $0 \leq f_1 \leq f_2 \leq \cdots \leq f_k$ and $0 \leq f'_1 \leq f'_2 \leq \cdots \leq f'_k$, such that $f_j \geq f'_j$ for all $j \leq k$. Let $v \geq 0$. Then*

$$W(v, \{f_1, \ldots, f_k\}) \leq W(v, \{f'_1, \ldots, f'_k\}). \tag{26}$$

Proof. Suppose that there are i future jobs at time v with release dates $\tilde{r}_1 < \tilde{r}_2 < \cdots < \tilde{r}_i$. We shall proceed by induction on i. First let $i = 0$. Then the completion times for the jobs with due dates $(f_1, \ldots, f_k)$ are the same as the completion times for the jobs with due dates $(f'_1, \ldots, f'_i)$, so $f_j \geq f'_j$ implies (26). Suppose now that the statement is true for $i - 1$ future jobs. Suppose that by time $\tilde{r}_1$ the jobs with due dates $(f_{j(1)}, \ldots, f_{j(l)})$ are completed at times $\tilde{C}_1, \ldots, \tilde{C}_l$. At exactly the same times jobs with due dates $(f'_{j(1)}, \ldots, f'_{j(l)})$ will be completed. Let f be the due date of the job that releases at $\tilde{r}_1$. Then

$$W(v, \{f_1, \ldots, f_k\}) = \max\{\tilde{C}_1 - f_{j(1)}, \ldots, \tilde{C}_l - f_{j(l)}, W(\tilde{r}_1, \{f_1, \ldots, f_k, f\} \setminus \{f_{j(1)}, \ldots, f_{j(l)}\})\}$$

and

$$W(v, \{f'_1, \ldots, f'_k\}) = \max\{\tilde{C}_1 - f'_{j(1)}, \ldots, \tilde{C}_l - f'_{j(l)}, W(\tilde{r}_1, \{f'_1, \ldots, f'_k, f\} \setminus \{f'_{j(1)}, \ldots, f'_{j(l)}\})\}.$$

At time $\tilde{r}_1$ there are only $i - 1$ future jobs, so by induction

$$W(\tilde{r}_1, \{f_1, \ldots, f_k, f\} \setminus \{f_{j(1)}, \ldots, f_{j(l)}\}) \leq W(\tilde{r}_1, \{f'_1, \ldots, f'_k, f\} \setminus \{f'_{j(1)}, \ldots, f'_{j(l)}\}),$$

hence (26) follows. □

Lemma 5. *Let $f_1, \ldots, f_k \geq 0$, and let $1 \leq j \leq k$. Let $v \geq 0$. Then*

$$W(v, \{f_1, \ldots, f_k\} \setminus \{f_j\}) \leq W(v, \{f_1, \ldots, f_k\}). \tag{27}$$

Proof. Assume that $f_1 \leq \cdots \leq f_k$. Let f be a due date of an additional job such that $f > f_k$, and also f is larger than the due date of every future job at time v. First we consider the situation in which instead of removing the job with due date f_j, we replace it by the job with due date f. By

Lemma 4 we have

$$W(v, \{f_1, \ldots, f_k, f\} \setminus \{f_j\}) \leq W(v, \{f_1, \ldots, f_k\}).$$

But since f is the largest due date, in the case when the due date f_j has been replaced by f, the completion times of all other jobs (those with due dates $f_1, \ldots, f_{j-1}, f_{j+1}, \ldots, f_k$) are the same as in the case when f_j has been removed. But then clearly

$$W(v, \{f_1, \ldots, f_k\} \setminus \{f_j\}) \leq W(v, \{f_1, \ldots, f_k, f\} \setminus \{f_j\}),$$

so (27) follows. □

Lemma 6. *Let Y be an $\mathcal{F}_{A(1)}$ measurable random variable, and $\mathcal{D} = \{f_1, \ldots, f_k\}$ a set of k positive numbers. Let A_p be an action time, V be the first jump of N after A_p, and $0 \leq i \leq k$. Then*

$$E\left[\max\left\{Y, V - f_i, W\left(V, \mathcal{D} \setminus \{f_i\}\right)\right\} | \mathcal{F}_V\right] = \tilde{F}_i(Y, V), \tag{28}$$

where

$$\tilde{F}_i(y, v) = E\left[\max\left\{y, v - f_i, v + W(V, \mathcal{D} \setminus \{f_i\}) - V\right\}\right].$$

Proof. $W\left(V, \mathcal{D} \setminus \{f_i\}\right) - V$ is independent of $\mathcal{F}_V$, whereas Y and V are measurable with respect to the same sigma field, so (28) follows. □

References

[1] Baptiste, P., Brucker, P., Knust, S., Timkovsky, V.G., 2004. Ten notes on equal-processing-time scheduling. Quart. J. Belgian, French, and Italian Oper. Res. Soc. 2, 111–127.

[2] Baruah, S., Bertogna, M., Buttazzo, G., 2015. Multiprocessor Scheduling for Real-Time Systems. Springer International Publishing.

[3] Brandwajn, A., Begin, T., 2017. Multi-server preemptive priority queue with general arrivals and service times. Performance Evaluation 115, 150–164.

[4] Buttazzo, G.C., 2011. Hard Real-Time Computing Systems. Springer US.

[5] Doytchinov, B., Lehoczky, J., Shreve, S., 2001. Real-time queues in heavy traffic with earliest-deadline-first queue discipline. Ann. Appl. Probab. 11, 332–378.

[6] Jacod, J., Shiryaev, A.N., 1987. Limit Theorems for Stochastic Processes. Springer Berlin Heidelberg.

[7] Kleinrock, L., 1976. Queueing Systems, Vol. II: Computer Applications. John Wiley & Sons.

[8] Lawler, E.L., Labetoulle, J., 1978. On preemptive scheduling of unrelated parallel processors by linear programming. J. Assc. Comput. Mach. 25, 612–619.

[9] Lehoczky, J.P., 1997. Using real-time queueing theory to control lateness in real-time systems. Assoc. Comput. Mach. Sigmetrics Perf. Eval. Rev. 25, 158–168.

[10] Leung, J.Y.T., Young, G.H., 1990. Preemptive scheduling to minimize mean weighted flow time. Inform. Process. Lett. 34, 47–50.

[11] Liu, Z., Nain, P., Towsley, D., 1995. Sample path methods in the control of queues. Queueing Sys. Appl. 21, 293–335.

[12] Liu, Z., Towsley, D., 1994. Effects of service disciplines in G/GI/s queueing systems. Ann. Oper. Res. 48, 401–429.

[13] Lushchakova, I.N., 2006. Two machine preemptive scheduling problem with release dates, equal processing times and precedence constraints. Eur. J. Oper. Res. 171, 107–122.

[14] Martel, C., 1982. Preemptive scheduling with release times, deadlines, and due times. J. Assc. Comput. Mach. 29, 812–829.

[15] Pinedo, M., Reed, J., 2013. The "least flexible job first" rule in scheduling and in queueing. Oper. Res. Lett. 41, 618–621.

[16] Pinedo, M.L., 2016. Scheduling. Springer International Publishing.

[17] Towsley, D., Panwar, S., 1992. Optimality of the stochastic earliest deadline policy for the G/M/c queue serving customers with deadlines, in: Second ORSA Telecomm. Conf., March 9-11, Operations Research Society of America, Boca Raton, FL.

[18] Wolff, R.W., 1970. Work-conserving priorities. J. Appl. Probab. 7, 327–337.