Journal of the Society of Korea Industrial and Systems Engineering
Vol. 35, No. 2, pp.173－180, June 2012.

충분히 이른 작업의 수를 최대화시키는 확률적 스케줄링 문제

최재영* · 김흥규**†

*동부CNI 연구소
**단국대학교 경영학부

Stochastic Scheduling Problems for Maximizing the Number of Early Enough Jobs

Jae Young Choi* · Heung-Kyu Kim**†

*R&D Center, Dongbu CNI
**School of Business, Dankook University

이 논문에서는 작업의 처리시간이 임의의 확률분포를 따르고 작업의 납기일이 작업마다 별개인 상황에서의 단일기계 스케줄링문제에 관하여 살펴본다. 이 때 충분히 이른 작업의 수를 최대화시키는 데에 관심을 둔다. 이러한 스케줄링문제를 풀기 위한 두 가지 알고리즘, 즉 이진정수계획모형과 스케줄링 규칙을 제안한다. 여기서 제안하는 스케줄링 규칙은 처리시간과 납기일이 확정적인 경우에 지연작업의 수를 최소화시켜주는 스케줄링을 제공하는 기존 알고리즘을 처리시간과 납기일이 확률적인 경우로 확장한 것이다. 다음으로 이진정수계획모형과 스케줄링규칙을 성과측면에서 비교한다. 그 결과 대부분의 경우에 스케줄링 규칙이 이진정수계획모형과 거의 같은 스케줄을 제공할 뿐만 아니라 컴퓨터자원을 매우 적게 소모한다.

Keywords : Stochastic Scheduling, Integer Programming, Scheduling Rule, Number of Early Enough Jobs

1. Introduction

In this paper, single machine stochastic scheduling problems are considered when processing times follow arbitrary distributions and due dates are distinct. There are many instances where the penalty for being late is the same regardless of how late it may be. For instance, any delay in the completion of any tasks required for a space launch will make the launch aborted. When problems fit this description, it is difficult to find an optimal schedule that maximizes the expected number of early jobs even if all the due dates are exponentially distributed. The problems of maximizing the expected number of early jobs have been investigated in many literatures. For concise review, see Choi and Kim [3]. Instead of maximizing the expected number of early jobs, a practitioner might be interested in maximizing the number of early enough jobs given a threshold α. A job is early enough if the probability of being early is greater than or equal to the threshold α This alternative objective might prove more amenable to efficient solutions.

Suppose there are n jobs to be scheduled. Each job i, $i=1,2,\cdots,n$, has its own processing time X_i which follows any non-negative valued distribution, and due date D_i which follows an exponential distribution with mean $1/\delta_i$. In addi-

논문접수일 : 2012년 03월 23일　　논문수정일 : 2012년 06월 06일　　게재확정일 : 2012년 06월 11일
† 교신저자 heungkyu@dankook.ac.kr

tion, suppose each processing time X_i is independent from the other processing times X_j, $j \neq i$. The objective is to maximize the number of early enough jobs.

This type of problems is considered in Balut [1], Katoh and Ibaraki [5], and Kise and Ibaraki [6]. In Balut [1], it is formulated as a threshold-constrained problem where the processing times are assumed to follow normal distributions and due dates are assumed to be distinct but deterministic. In the formulation, jobs are ordered in the increasing order of due dates, i.e., $D_1 \leq D_2 \leq \cdots \leq D_n$. The threshold-constrained formulation is as follows.

$$\begin{aligned} & Max. \sum_{j=1}^{n} y_j && (1) \\ & s.t.\ \Pr\left\{\sum_{i=1}^{j} X_i y_i \leq D_j\right\} \geq \alpha, \quad j=1,2,\cdots,n \\ & \qquad y_j = 0 \text{ or } 1, \qquad\qquad j=1,2,\cdots,n \end{aligned}$$

where $y_j = 1$ if job j is included in the schedule and $y_j = 0$ otherwise. By the threshold-constraints, it is assured that if a job is included in the schedule, the probability of the job being early is greater than or equal to the threshold α.

Unfortunately, the above formulation has some drawbacks. First, it is implicitly assumed by the threshold-constraints that the jobs selected for processing are processed in the increasing order of due dates. There is no discussion on this in Balut [1]. Second, the processing times are assumed to follow normal distributions while processing times can not take negative values. Finally, even if the jobs are not selected for processing, these jobs must satisfy the threshold-constraints. However, this drawback can be easily corrected by changing the right-hand sides of the threshold-constraints from α to αy_j.

Using the above formulation, a polynomial time algorithm is proposed in Balut [1]. However, there shows a counter-example for which the algorithm is unable to find an optimal schedule in Kise and Ibaraki [6]. Moreover, it is also shown that the problem reduces to a knapsack problem and thus the problem is NP-Complete. For a special case where $E[X_i] < E[X_j]$ implies $V(X_i) \leq V(X_j)$, a polynomial time algorithm is proposed in Kise et al. [7].

The rest of this paper is organized as follows. In Section 2, it is shown that a lemma proposed in Jackson [4] still holds if the due dates are exponentially distributed. Utilizing this lemma, it is shown that, in optimal schedules, the jobs selected for processing are processed in the increasing order of expected due dates. In Section 3, a binary integer programming formulation based on the characteristics of optimal schedules is proposed. A new scheduling rule, based on an algorithm in Moore [8], which solves the problem in some cases, is also proposed. In Section 4, the binary integer programming formulation is compared with the new scheduling rule in terms of performances. Final remarks are addressed in Section 5.

2. Characteristics of Optimal Schedules

A deterministic scheduling problem is investigated in Jackson [4]. In Jackson [4], it is shown that the earliest due date schedule has no late jobs if there exists a schedule having no late jobs. In our setting where processing times are represented by any independently non-negative valued random variables and due dates are represented by independently and exponentially distributed random variables, the lemma in Jackson [3] still holds.

Lemma 1

Let X_i, D_i, and $C_i(\pi)$ be the processing time, due date, and completion time for job i in schedule π, respectively. Given a set of jobs $\{1, 2, \cdots, n\}$ with any independently non-negative valued processing times and independently exponential due dates, there exists a schedule having no late jobs if and only if there are no late jobs in the schedule obtained by ordering the jobs in the increasing order of expected due dates.

Proof

It is sufficient to show that for a schedule π having no late jobs, the jobs in schedule π can be reordered in the increasing order of expected due dates. Suppose that, in schedule π there are two adjacent jobs j and i where job j is processed prior to job i and $\delta_j \leq \delta_i$, i.e., job i has a stochastically smaller due date than job j does ($D_i \leq_{st} D_j$). Let $\pi = (\pi_1, j, i, \pi_2)$ denote the schedule. The assumption implies

$$\begin{aligned} & \Pr\{C_i(\pi) \leq D_i\} \\ & = \Pr\left\{\sum_{k \in \pi_1} X_k + X_j + X_i \leq D_i\right\} \\ & = \prod_{k \in \pi_1} f_{ki} f_{ji} f_{ii} \\ & \geq \alpha \end{aligned}$$

and

$$\begin{aligned}&\Pr\{C_j(\pi) \le D_j\}\\ &= \Pr\left\{\sum_{k \in \pi_1} X_k + X_j \le D_j\right\}\\ &= \prod_{k \in \pi_1} f_{ki} f_{jj}\\ &\ge \alpha\end{aligned}$$

where $f_{ki} = \Pr\{X_k \le D_i\}$. By interchanging the two jobs i and j, the probability of job i being early is $\prod_{k \in \pi_1} f_{ki} f_{ii} \ge \prod_{k \in \pi_1} f_{ki} f_{ji} f_{ii} \ge \alpha$ since $0 < f_{ji} \le 1$. For job j, the probability of being early after the interchange is $\prod_{k \in \pi_1} f_{kj} f_{ij} f_{jj} \ge \prod_{k \in \pi_1} f_{ki} f_{ji} f_{ii} \ge \alpha$ since $\Pr\{X \le D_j\} \ge \Pr\{X \le D_i\}$ for any non-negative valued random variable X.

Although the lemma in Jackson [4] deals with the schedules without late jobs, it provides a basis for building the characteristics of optimal schedules. This section mimics the theoretical results in Moore [8]. In a deterministic setting, it is shown that, in an optimal schedule, early jobs are processed in the increasing order of due dates.

For a given schedule π, two ordered sets, E and L of early enough and late enough jobs, respectively, are defined as $E = \{i : \Pr\{C_i(\pi) \le D_i\} \ge \alpha\}$ and $L = \{i : \Pr\{C_i(\pi) \le D_i\} < \alpha\}$, where, of course, a job is late enough if the probability of being early is less than the threshold α. The elements in E and L are ordered in the increasing order of expected due dates in schedule π.

Lemma 2

Any optimal schedule π is equivalent to schedule $\pi^* = (E, L)$ in terms of the number of late enough jobs.

Proof

Let $C_{\pi(k)} = C_{\pi(k)}(\pi)$ for simplicity where $\pi(k)$ is the kth job in schedule π. Suppose an optimal schedule π is not in the form of (E, L). Then, there must be at least two adjacent jobs, $\pi(k)$ and $\pi(k+1)$ in π for $k \ge 1$ such that $\Pr\{C_{\pi(k)} \le D_{\pi(k)}\} < \alpha$ and $\Pr\{C_{\pi(k+1)} \le D_{\pi(k+1)}\} \ge \alpha$. If we interchange $\pi(k)$ and $\pi(k+1)$, we have new completion times $C'_{\pi(k)} = C_{\pi(k+1)} + X_{\pi(k)}$ and $C'_{\pi(k+1)} = C_{\pi(k-1)} + X_{\pi(k+1)}$ with $C_{\pi(0)} = 0$. In the new schedule, the probabilities of jobs $\pi(k)$ and $\pi(k+1)$ being early, respectively, are

$$\begin{aligned}&\Pr\{C'_{\pi(k)} \le D_{\pi(k)}\}\\ &= \Pr\{C_{\pi(k+1)} + X_{\pi(k)} \le D_{\pi(k)}\}\\ &\le \Pr\{C_{\pi(k+1)} \le D_{\pi(k)}\}\\ &< \alpha\end{aligned}$$

$$\begin{aligned}&\Pr\{C'_{\pi(k+1)} \le D_{\pi(k+1)}\}\\ &= \Pr\{C_{\pi(k-1)} + X_{\pi(k+1)} \le D_{\pi(k+1)}\}\\ &\ge \Pr\{C_{\pi(k+1)} \le D_{\pi(k+1)}\}\\ &\ge \alpha\end{aligned}$$

Thus, repeated interchanges in this manner to produce an (E, L) schedule does not change the number of late enough jobs.

Let E_D be the set of jobs in E ordered in the increasing order of expected due dates. Then, the following corollary must hold.

Corollary 1

A schedule (E, L) is equivalent to schedule (E_D, L) in terms of the number of late enough jobs.

Corollary 1 plays an important role in our problem formulation. The only thing left to be considered is to decide which jobs should be in the set of early enough jobs since, in an optimal schedule, early enough jobs are processed in the increasing order of expected due dates.

3. Formulation and Scheduling Rule

Formulation In the following, jobs are indexed in increasing order of their expected due dates, i.e., $\delta_1 \ge \delta_2 \ge \cdots \ge \delta_n$. Let $J = \{1, 2, \cdots, n\}$ denote the set of all the jobs. The key idea of Corollary 1 is captured in the following formulation which selects jobs to be early enough in an optimal schedule.

$$\begin{aligned}&Max. \sum_{j=1}^{n} y_j && (2)\\ &s.t.\ \Pr\left\{\sum_{i=1}^{j} X_i y_i \le D_j\right\} \ge \alpha y_j, && \text{for } all\ j \in J\\ &\qquad y_j = 0 \text{ or } 1, && \text{for } all\ j \in J\end{aligned}$$

Note that problem formulation (2) differs from problem formulation (1) in that, in problem formulation (2), indication variable y_j is added on the right-hand side of the threshold-constraints in problem formation (1). It turns out that the added indication variable y_j does not need to be shown in Lemma 3 below. Due to the exponential nature of the due dates, each constraint in formulation (2) above can be simplified as a product form $\prod_{i=1}^{j}(f_{ij})^{y_i} \geq \alpha y_j$, where again $f_{ij} = \Pr\{X_i \leq D_j\}$. If we take logarithms on both sides of those constraints, problem formulation (2) becomes a binary integer programming formulation as follows.

$$\begin{aligned} & Max. \sum_{j=1}^{n} y_j && (3) \\ & s.t.\ \ \log y_j + \sum_{i=1}^{j} y_i p_{ij} \leq \beta, \text{ for } all\, j \in J \\ & \qquad y_j = 0 \text{ or } 1, \qquad \text{for } all\, j \in J \end{aligned}$$

where $p_{ij} = -\log f_{ij} = -\log \Pr\{X_i \leq D_j\}$ and $\beta = -\log\, \alpha$.

Let us consider another binary integer programming formulation that is very similar to problem formulation (3).

$$\begin{aligned} & Max. \sum_{j=1}^{n} y_j && (4) \\ & s.t.\ \ \sum_{i=1}^{j} y_i p_{ij} \leq \beta, \qquad \text{for } all\, j \in J \\ & \qquad y_j = 0 \text{ or } 1, \qquad \text{for } all\, j \in J \end{aligned}$$

The only thing that makes problem formulation (4) differ from problem formulation (3) is no existence of $\log y_j$ in each constraint in problem formulation (4). However, it can be shown that an optimal schedule to problem formulation (4) is also an optimal schedule to problem formulation (3).

Lemma 3

An optimal schedule to problem formulation (4) is also an optimal schedule to problem formulation (3).

Proof

Suppose y^* is an optimal schedule to problem formulation (4) and not optimal to problem formulation (3). Let $E^* = \{i \in J \colon y_i^* = 1\}$. Then, there must be at least one job j such that $y_j = 1$ and $y_j \not\in E^*$, but satisfies all the constraints in problem formulation (3). Therefore, we have $\log y_j + \sum_{k=1}^{j-1} y_k^* p_{kj} + y_j p_{jj} \leq \beta$, which means that, by making $y_j^* = 1$, we have the corresponding constraint in problem formulation (4) also satisfied. This contradicts the assumption that y^* is optimal.

Order Invariant Processing Times Problem formulation (4) can be expressed in a matrix form as follows.

$$\begin{aligned} & Max. \, ey \\ & s.t.\ \ Py \leq \beta \end{aligned}$$

where e is a row vector of ones and the elements of the lower triangular matrix P are $p_{ij} = -\log \Pr\{X_i \leq D_j\}$ for $i \geq j$ and 0 otherwise.

Indeed problem formulation (4) can be regarded as a capital allocation problem, which does not have a polynomial time algorithm unless it can be solved by a linear programming relaxation. For details, see Nemhauser and Ullmann [9]. However, for a special case where P is 'order invariant', the problem can be solved efficiently with the complexity of sorting algorithms. Before we propose an algorithm, we define 'order invariant'.

Definition 1

A matrix P is order invariant if $0 < p_{ik} < p_{jk}$ implies $p_{i,k+1} < p_{j,k+1}$ for all i, j, and k.

If matrix P is order invariant, the scheduling rule, shown in <Figure 1>, can be used to find an optimal schedule to maximize the number of early enough jobs in polynomial time $O(n \log n)$.

Step 1 : Sort the jobs in the increasing order of expected due dates, i.e., in the decreasing order of δ.
Step 2 : Set $E = \{1\}$ and $L = J$. Set $i = 1$.
Step 3 : If $\prod_{j \in E} f_{ji} < \alpha$, or equivalently, $\sum_{j \in E} p_{ji} > \beta$, then find a job s such that $f_{si} = \min_{j \in E} f_{ji}$ or $p_{si} = \max_{j \in E} p_{ji}$. Set $E = E - \{s\}$ and $L = L \cup \{s\}$.
Step 4 : $i = i + 1$. If $i > n$, terminate. Otherwise, $E = E \cup \{i\}$, and return to Step 3.

<Figure 1> New Scheduling Rule

Theorem 1

If P is order invariant, then the new scheduling rule guarantees an optimal schedule to problem formulation (4).

Proof

It is sufficient to show that if a job s is found in Step 3, then $s \in L$ for some optimal schedule for the jobs $\{1, \cdots, i\}$ in the current sequence. Consider an optimal schedule of the form (E_D, L) in the current sequence. If $s \in L$, we are done. Hence, assume $s \in E_D$ and E_D is of the form : $(i_1, \cdots, i_r, \cdots, i_t)$

and

$$\delta_{i_1} > \cdots > \delta_{i_r} > \delta_{i_{r+1}} > \cdots > \delta_{i_t}$$

where

$$i_r = s.$$

Thus, $\sum_{j \in \{j_1, \cdots, j_k\}}^{i} p_{ji}$ is minimal over the class of all subsets of k jobs from the current sequence having values less than or equal to β. Hence, since job s is late enough in the sequence $(j_1, \cdots, j_k, s)$, it will be late enough in all due date ordered sequences in which it is the $(k+1)$th job. But $i_r = s$ is early in the schedule (E, L) and therefore r must be strictly less than $(k+1)$. Further

$$\{i_{r+1}, \cdots, i_t\} \cap \{j_1, \cdots, j_k\} = \varnothing,$$

since $\delta_{j_i} > \delta_s$, $i = 1, \cdots, k$, and $\delta_{i_j} < \delta_s$, $j = r+1, \cdots, t$. Hence, for a member of $\{j_1, \cdots, j_k\}$ to be on time in the schedule (E_D, L), it must be a member of $\{i_1, \cdots, i_{r-1}\}$. Consequently, there are jobs in the set $\{j_1, \cdots, j_k\}$ that are members of the set L in the schedule (E_D, L) since $r < k+1$. Another optimal schedule (E_D^*, L^*) can now be found by replacing the initial sequence $(i_1, \cdots, i_r)$ in the schedule (E_D, L) with the first r jobs from the set $\{j_1, \cdots, j_k\}$. The job $i_r = s$ is now a member of L^* for the optimal schedule (E_D^*, L^*).

What kind of processing times make P order invariant, or equivalently, what conditions for processing times are required, for the algorithm to work remains an open question. To deal with this question, we should discuss the concept of stochastic ordering first.

A random variable X_i is said to be stochastically less than or equal to a random variable X_j in an increasingly concave ordering sense if and only if $E[\varnothing(X_i)] \le E[\varnothing(X_j)]$ for all the increasingly concave functions $\varnothing(\cdot)$. This relationship is denoted by $X_i \le_{icv} X_j$. The increasingly concave ordering is weaker than stochastic ordering, that is, if $X_i \le_{st} X_j$, then $X_i \le_{icv} X_j$. For details, see Chang et al. [2] or Ross [9].

The condition of 'order invariant' is automatically satisfied when all the processing times can be ordered in an increasingly concave ordering sense and due dates have concave distributions as when they are exponentially distributed.

Lemma 4

If $X_i \le_{icv} X_j$, then $\Pr\{X_i \le D\} \ge \Pr\{X_j \le D\}$ for any non-negative valued random variable D whose distribution function, $H(x)$, is concave.

Proof

$\Pr\{X \le D\} = 1 - \Pr\{X > D\} = 1 - \int_0^\infty H(x) f(x) dx$, where $f(x)$ is the probability density function of X. Since $H(x)$ is an increasingly concave function and $X_i \le_{icv} X_j$, we have $\int_0^\infty H(x) f_i(x) dx \le \int_0^\infty H(x) f_j(x) dx$. This leads to $\Pr\{X_i \le D\} \ge \Pr\{X_j \le D\}$.

Note that an exponential random variable has a concave distribution function. Therefore, when all the processing times can be ordered in an increasingly concave ordering sense and due dates are exponentially distributed, the corresponding matrix becomes order invariant.

Arbitrary Processing Times The scheduling rule in <Figure 1> cannot be applied to a problem with general processing times. This is because it can not be guaranteed that the corresponding P is order invariant. It can be shown that the probability of being early for any non-negative valued random variable X, $\Pr\{X \le D(\delta)\}$, is decreasingly convex in δ, where $D(\delta)$ is an exponential random variable with rate δ.

Proposition 1

The earliness probability $\Pr\{X \le D(\delta)\}$ is decreasingly convex in δ for any non-negative valued random variable X.

Proof

Let $f(x)$ denote the probability density function of X and L_X denote the Laplace-Stieltjes transform of X. Since $\Pr X \le D(\delta)\} = E[e^{-\delta X}] = L_X(\delta)$, the probability of being early is the Laplace-Stieltjes transform of X at δ or equivalently the Laplace transform of its probability density function $f(x)$. It is known that the inverse of the first derivative of L_X is $-xf(x) < 0$ and the inverse of the second derivative of L_X is $x^2 f(x) > 0$. Hence, the first derivative of the probability is the Laplace transform of $-xf(x)$ which is negative, and the second derivative of the probability is the Laplace transform of $x^2 f(x)$ which is positive. This completes the proof.

From the fact that earliness probability $\Pr\{X \le D(\delta)\}$ is convex in δ, it can be deduced that for any two non-negative random variables, X and Y, there is at most one δ', $0 < \delta' < \infty$, such that $E[e^{-\delta' X}] = E[e^{-\delta' Y}]$. This, in turn, means that the order of the probabilities can be reversed at some rate δ. This fact makes it impossible to guarantee order invariant property for general processing times. Hence, in this case, we need to solve problem formulation (4) instead.

4. Computational Experiments

So far, two methods, for maximizing the number of early enough jobs when processing times are arbitrary distributed and due dates are exponentially distributed, are considered. One is a binary integer programming formulation, which guarantees an optimal solution, however, is computationally infeasible for problems of moderate sizes. The other is a scheduling rule, which guarantees an optimal solution only in special cases, however, is computationally feasible for problems of any sizes.

In this section, the two methods are compared with each other in terms of performances when either job processing times or due dates are generally distributed. Earlang random variables for either processsing times or due dates are chosen because they are non-negative valued and can have many different shapes with varying two parameters.

CPLEX is used for solving the binary integer program. Both methods are programmed in ANSI C programming language and run on Sun Unix Systems.

4.1 Number of Early Enough Jobs

Each job i is assumed to have a processing time which follows an Erlang distribution with k_i phases and a λ_i phase rate. A set of Erlang random variables is not always order invariant. If a set of processing times are not order invariant, the binary integer programming formulation in Section 3 should be employed to obtain an optimal schedule.

Data Preparation In this experiment, the values for two factors, the number of jobs and the minimum survivability among the jobs, varied. The survivability of job i is defined as the probability of being early if the job is processed first, i.e., $\Pr\{X \le D(\delta)\}$. It is handy to make up data sets according to their minimum survivabilities in order for us to easily look at the behavior of the two approaches based on each level of minimum survivability.

<Table 1> Number of Early Enough Jobs

# of Job	Min. Surv.	Avg. Early Jobs		Avg. CPU Times	
		Binary Integer Prog.	Sched. Rule	Binary Integer Prog.	Sched. Rule
10	0.50	4.5200	4.5200	0.0153	0.0000
10	0.60	5.2400	5.2400	0.0163	0.0003
10	0.70	6.2733	6.2733	0.0157	0.0000
10	0.80	7.5733	7.5733	0.0140	0.0000
10	0.90	8.9933	8.9933	0.0120	0.0000
10	0.95	9.8267	9.8267	0.0053	0.0000
Avg.		7.0711	7.0711	0.0131	0.0001
30	0.50	9.1400	9.1400	0.1033	0.0000
30	0.60	11.5133	11.5133	0.1340	0.0003
30	0.70	13.2200	13.2200	0.1597	0.0007
30	0.80	16.4667	16.4667	0.1510	0.0000
30	0.90	23.0867	23.0867	0.0967	0.0007
30	0.95	27.1333	27.1333	0.0443	0.0003
Avg.		16.7600	16.7600	0.1148	0.0003
50	0.50	13.1867	13.1867	0.2923	0.0007
50	0.60	15.1400	15.1400	0.3200	0.0023
50	0.70	19.1400	19.1400	0.3903	0.0007
50	0.80	23.9133	23.9133	0.3930	0.0017
50	0.90	33.0200	33.0200	0.3000	0.0023
50	0.95	42.4733	42.4733	0.1457	0.0007
Avg.		24.4789	24.4789	0.3069	0.0014
70	0.50	15.9600	15.9600	0.5630	0.0017
70	0.60	19.8000	19.8000	0.6883	0.0033
70	0.70	21.9933	21.9867	0.8107	0.0020
70	0.80	28.5133	28.5133	0.8990	0.0010
70	0.90	39.8267	39.8267	0.7113	0.0027
70	0.95	54.7333	54.7333	0.3850	0.0027
Avg.		30.1378	30.1367	0.6762	0.0022
90	0.50	20.0733	20.0733	0.8467	0.0023
90	0.60	21.6467	21.6467	1.1863	0.0023
90	0.70	26.8467	26.8400	1.5040	0.0043
90	0.80	33.5333	33.5333	1.7393	0.0013
90	0.90	47.5467	47.5467	1.6663	0.0040
90	0.95	65.5867	65.5867	0.8703	0.0060
Avg.		35.8722	35.8711	1.3022	0.0034
100	0.50	20.3533	20.3533	1.1857	0.0047
100	0.60	23.4133	23.4133	1.4367	0.0040
100	0.70	27.3267	27.3267	1.8953	0.0027
100	0.80	34.8800	34.8800	2.1163	0.0037
100	0.90	52.3333	52.3333	2.0040	0.0037
100	0.95	71.2867	71.2867	1.7427	0.0063
Avg.		38.2656	38.2656	1.7301	0.0042
Total Avg.		25.4309	25.4306	0.6906	0.0019

The number of jobs varied over 10, 30, 50, 70, 90, and 100. The minimum survivability varied over 0.5, 0.6, 0.7, 0.8, 0.9, and 0.95. For each combination of job size and minimum survivability, 30 random samples from Erlang processing times and exponential due dates are generated. The number of phases for an Erlang random variable is randomly drawn from a discrete uniform distribution $U(1, 8)$. The rate of each phase is also randomly generated from a uniform distribution $U(0.1, 2.0)$. The value of survivability for a given job is randomly drawn from a uniform distribution $U(p', 1.0)$, where p' is the specified minimum survivability. Once parameters for the processing times and survivability are given, the rate of the exponentially distributed random variable corresponding to each expected due date can be automatically determined. That is, for an Erlang random variable X_i and an exponential due date $D(\delta_i)$, δ_i can be easily determined for a given survivability p_i from $\Pr\{X_i \le D(\delta_i)\} = \left[\frac{\lambda_i}{\lambda_i+\delta_i}\right]^k$.

Results There are 30 random samples generated for each combination of job size and minimum survivability as previously described. For each data set, the performance of the binary integer programming formulation is compared to that of the scheduling rule with various thresholds, 0.5, 0.6, 0.7, 0.8, and 0.9. The average number of early jobs and the average CPU times of the binary integer programming formulation versus those of the scheduling rule are shown in <Table 1>. There is only a little difference (< 0.001%) in terms of the maximum number of early enough jobs, while CPU times of the binary integer programming formulation surpass those of the scheduling rule. Thus, the scheduling rule achieves near optimality in only a fraction of the time.

4.2 Expected Number of Early Jobs

Even if due dates are exponentially distributed, there is no known solution method to get the maximum expected number of early jobs except for total enumeration when processing times have general distributions. The binary integer programming formulation proposed in this paper is used to approximate the maximum expected number of early jobs. The main idea is that the binary integer programming formulation is solved for a given set of jobs with many threshold points, for example, 100 points between 0.01 and 0.99. For each threshold point, early enough jobs will be selected and the selected jobs will be processed in the increasing order of due dates. The late jobs, i.e., the jobs not selected, will be processed in a manner of 'shortest processing time' first after the early enough jobs are processed. Once the sequence is determined, the expected number of early jobs is examined for the given threshold point. The sequence is obtained that gives the maximum expected number of early jobs throughout the entire threshold points.

The data sets in the previous subsection are used again for comparing the performances of the binary integer programming formulation to those of the scheduling rule. The scheduling rule runs much faster (9.97 times on the average) than the binary integer programming formulation while the performances of approximation with the scheduling rule are 99.91% of those of the binary integer programming formulation on average. The results are shown in <Table 2>.

<Table 2> Expected Number of Early Jobs

# of Job	Min. Surv.	Avg. Early Jobs		Avg. CPU Times	
		Binary Integer Prog.	Sched. Rule	Binary Integer Prog.	Sched. Rule
10	0.50	5.0365	5.0330	0.3227	0.2770
10	0.60	5.7031	5.6997	0.3057	0.2740
10	0.70	6.5440	6.5430	0.2713	0.2720
10	0.80	7.4927	7.4952	0.2490	0.2710
10	0.90	8.6046	8.6046	0.2103	0.2673
10	0.95	9.3165	9.3171	0.1877	0.2670
Avg.		7.1162	7.1154	0.2578	0.2714
30	0.50	9.1670	9.1270	2.4373	0.5000
30	0.60	11.0818	11.0568	2.3797	0.4620
30	0.70	12.6612	12.6391	2.2277	0.4413
30	0.80	15.7160	15.6871	1.7767	0.4083
30	0.90	21.2300	21.2361	1.1967	0.3663
30	0.95	25.1553	25.1572	0.7870	0.3467
Avg.		15.8352	15.8172	1.8009	0.4208
50	0.50	12.5241	12.5052	6.9377	0.9940
50	0.60	14.2358	14.1934	6.5213	0.9083
50	0.70	17.5487	17.5486	5.9150	0.7960
50	0.80	21.4537	21.3969	5.1980	0.7183
50	0.90	29.5748	29.5756	3.5753	0.5970
50	0.95	38.3053	38.3063	2.3273	0.5040
Avg.		22.2737	22.2543	5.0791	0.7529
70	0.50	14.9072	14.8775	14.3180	1.8317
70	0.60	17.9152	17.8629	14.9523	1.6327
70	0.70	19.7764	19.7338	15.0777	1.5290
70	0.80	25.3963	25.3863	12.5207	1.2730
70	0.90	34.8381	34.8052	9.0267	1.0053
70	0.95	48.3340	48.3339	6.2100	0.8060
Avg.		26.8612	26.8333	12.0176	1.3463
90	0.50	17.9018	17.8608	26.0430	2.8550
90	0.60	19.4192	19.3918	26.0847	2.7967
90	0.70	23.6239	23.5877	28.6757	2.4357
90	0.80	29.4188	29.3780	24.4007	2.0683
90	0.90	41.0114	40.9683	17.4287	1.5663
90	0.95	56.8936	56.8946	12.4313	1.2197
Avg.		31.3781	31.3469	22.5107	2.1570
100	0.50	18.2140	18.2255	37.1717	3.6803
100	0.60	20.8718	20.8525	36.5447	3.3907
100	0.70	24.0271	23.9963	40.9517	3.0857
100	0.80	30.3401	30.2796	40.0807	2.6647
100	0.90	44.4385	44.4130	32.2067	1.8997
100	0.95	60.9822	60.9773	20.3840	1.4537
Avg.		33.1456	33.1240	34.5566	2.6958
Total Avg.		22.7684	22.7485	12.7038	1.2740

5. Final Remarks

After the failure of the algorithms in Balut [1] and Kise and Ibaraki [6] for the threshold-constrained stochastic scheduling problem, the problem has never been addressed so far to the best of our knowledge, which is confirmed by the remarks in Choi and Kim [3] and in Sterna [11]. In this paper, it is shown that the lemma in Jackson [4] still holds in a stochastic setting where all the due dates are exponentially distributed. Based on this observation, the binary integer programming formulation for maximizing the number of early enough jobs is derived. Moreover, a new scheduling rule is developed for obtaining a set of early enough jobs to be processed for a given threshold. It is also shown that if processing times meet order invariant condition, the scheduling rule finds optimal schedules. Otherwise, the problem can be solved by the binary integer programming formulation.

When processing times are Erlang distributed, the scheduling rule is compared to the binary integer programming formulation. The scheduling rule runs much faster than the binary integer programming formulation while there is almost no loss in solution quality in terms of the average number of early jobs.

The binary integer programming formulation is also applied to approximate the maximum expected number of early jobs. Just like the previous result, the scheduling rule runs faster while its performance is comparable to the binary integer programming formulation.

References

[1] Balut, S. J.; "Scheduling to Minimize the Number of Late Jobs When Set-Up and Processing Times are Uncertain," *Management Science*, 19 : 1283-1288, 1973.

[2] Chang, C.-S., Shanthikumar, J. G., and Yao, D. D.; "Stochastic Convexity and Stochastic Majorization," *Stochastic Modeling and Analysis of Manufacturing Systems*, Springer-Verlag, New York, 1994.

[3] Choi, J. and Kim, H.; "A Review on Scheduling Problems for Minimizing the Number of Late Jobs," *Korean Production and Operations Management Society*, 22 : 159-175, 2011.

[4] Jackson, J. R.; "Scheduling a Production Line to Minimize Maximum Tardiness," *Research Report*, 43 : UCLA, 1955.

[5] Katoh, N. and Ibaraki, T.; "A Polynomial Time Algorithm for A Chance-Constrained Single Machine Scheduling Problem," *Operations Research Letters*, 2 : 62-65, 1983.

[6] Kise, H. and Ibaraki, T.; "On Balut's Algorithm and NP-Completeness for a Chance-Constrained Scheduling Problem," *Management Science*, 29 : 384-388, 1983.

[7] Kise, H., Shiomi, A., Uno, M., and Chao, D.-S.; "An Efficient Algorithm for a Chance-Constrained Scheduling Problem," J. *Operations Res. Soc. Japan*, 25, 193-203, 1982.

[8] Moore, M. J.; "An N Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs," *Management Science*, 15 : 102-109, 1968.

[9] Nemhauser, G. L. and Ullmann Z.; "A Note on the Generalized Lagrange Multiplier Solution to an Integer Programming Problem," *Operations Research*, 16, 450-453, 1968.

[10] Ross, S. M.; Stochastic Processes, John Wiley & Sons, second edition, 1996.

[11] Sterna, M.; Late work scheduling in shop systems, Publishing House of Poznan University of Technology, 2006.