Minimizing the Number of Late Jobs with Release Time Constraint

Jianzhong Du, Joseph Y-T. Leung and C.S. Wong[1]

Computer Science Program
University of Texas at Dallas
Richardson, TX 75083

Abstract. We consider the problem of preemptively scheduling a set of N independent jobs with release times on $m \geq 1$ identical machines so as to minimize the number of late jobs. For a single machine, Lawler has given an $O(N^5)$ time algorithm for finding a schedule with the minimum number of late jobs. However, for fixed $m \geq 2$, the question of whether the problem can be shown to be solvable in polynomial time or shown to be NP-hard remained open over the last decade. In this paper we answer this question by showing that it is NP-hard for every fixed $m \geq 2$.

1. Introduction

A fundamental problem in deterministic scheduling theory is that of scheduling a set $JS = \{J_1, J_2, \ldots, J_N\}$ of N independent jobs with release times on $m \geq 1$ identical machines so as to minimize the number of *late jobs*. Each job J_i has associated with it a *release time* $r(J_i)$, *a processing time* $p(J_i)$, and a *due date* $d(J_i)$. A job cannot start until its release time and it is expected to be completed by its due date. With respect to a schedule S, a job is said to be *late* if it is completed after its due date; otherwise, it is said to be *on-time*. Our goal is to find a schedule such that the number of late jobs is minimized; or equivalently, the number of on-time jobs is maximized. This problem has received considerable interest since Moore [10] gave an $O(N \log N)$ time algorithm, more than two decades ago, for a special case of the problem in which $m = 1$ and all release times are equal. In this paper we consider the preemptive version of this problem for $m \geq 2$.

In nonpreemptive scheduling, it is well known [2] that the problem of deciding if a given set of jobs is *feasible* on a single machine is strongly NP-complete; i.e., each job is scheduled between its release time and its due date. This implies that the problem of minimizing the number of late jobs is strongly NP-hard for every fixed $m \geq 1$. However, special cases of this problem have been shown to be solvable in polynomial time. As mentioned earlier, the special case where $m = 1$ and all release times are equal can be solved by an $O(N \log N)$ time algorithm due to Moore. Sidney [11] later extended Moore's algorithm to allow specification of certain jobs required to be on-time. Kise et al. [3] have given an $O(N^2)$ time algorithm for the more general case where the jobs have similarly ordered release times and due dates; i. e., there is a labeling of the jobs such that $r(J_1) \leq r(J_2) \leq \cdots \leq r(J_N)$ and $d(J_1) \leq d(J_2) \leq \cdots \leq d(J_N)$. Most recently, Lawler [7] gave an improved algorithm for the special case studied by

[1]Research supported in part by the ONR grant N0001487-K-0833.

Kise et al. with a running time of $O(N \log N)$. If all release times are equal, the problem of minimizing the number of late jobs is NP-hard for each fixed $m \geq 2$; it becomes strongly NP-hard for arbitrary m [2]. We can complicate the problem, for example, by giving a weight to each job or adding precedence constraints among the jobs. We refer the reader to [6,7,8] for results on these complications.

From the above discussions, we can see that the complexity issues for nonpreemptive scheduling have been well solved. This is not the case for preemptive scheduling. Unlike nonpreemptive scheduling, the feasibility problem for preemptive scheduling can be solved in polynomial time, even for an arbitrary number of unrelated machines [9]. Thus, one would expect that more algorithmic results exist for preemptive scheduling, and indeed this is so. For example, Lawler [7] gave a dynamic programming algorithm with time complexity $O(N^5)$ for a single machine. Furthermore, it is well known that for a single machine, there is no advantage to preempt a job if the jobs have similarly ordered release times and due dates. Thus, all of the corresponding results for nonpreemptive scheduling mentioned above remain valid for preemptive scheduling. When all release times are equal and weights are present, the dynamic programming algorithms due to Lawler [5] can solve the problem for m uniform machines in $O(N^2W^2)$ time for $m = 2$ and in $O(N^{3m-5}W^2)$ time for $m > 2$, where W is the total weight of the jobs. Hence, for a fixed number of uniform machines and equal release times, the problem of minimizing the number of late jobs is solvable in polynomial time. Unfortunately, however, Lawler [6] also showed that the problem becomes NP-hard for an arbitrary number of identical machines.

It is somewhat unexpected that little was known about preemptive scheduling for fixed $m \geq 2$ and unequal release times. As noted in [4], the question of whether the problem can be shown to be NP-hard or shown to be solvable in polynomial time for fixed $m \geq 2$ remained open. In this paper we answer this question by showing that the problem is NP-hard for every fixed $m \geq 2$.

We now introduce some notations and conventions that will be used throughout the paper. Let S be a schedule. The symbol $NLJ(S)$ denotes the number of late jobs in S. If a job is late in S, it is clearly immaterial where it is scheduled in S. Without loss of generality, we may assume that all late jobs are scheduled after the latest due date. In the remainder of this paper, we will only be concerned with how to schedule the on-time jobs, assuming that all late jobs are scheduled after the latest due date. Also, in our discussions, we say that a machine is *idle* during some time interval in S if no jobs are processed by the machine during that interval in S.

2. NP-hardness Proof

In this section we show that the problem of finding a preemptive schedule with the minimum number of late jobs on m identical machines is NP-hard for every

fixed $m \geq 2$. We show this result by proving the corresponding decision problem to be NP-complete. The decision problem with parameter m can be defined as follows.

NLJP(m). *Given an integer K and a set $JS = \{J_1, J_2, \ldots, J_N\}$ of N independent jobs, where each job J_i has an integer release time $r(J_i)$, an integer processing time $p(J_i)$ and an integer due date $d(J_i)$, is there a preemptive schedule S for JS on m identical machines such that $NLJ(S) \leq K$?*

We first show that the $NLJP(2)$ problem is NP-complete; the proof will then be generalized to $m > 2$ at the end of the section. To show that the $NLJP(2)$ problem is NP-complete, we reduce to it a restricted version of the NP-complete Even-Odd Partition problem [2]. The Even-Odd Partition problem and the Restricted Even-Odd Partition problem are stated as follows.

Even-Odd Partition. *Given a set $A = \{a_1, a_2, \ldots, a_{2n}\}$ of $2n$ positive integers, where $a_i < a_{i+1}$ for each $1 \leq i \leq 2n$, is there a partition of A into two subsets A_1 and A_2 such that $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2}$ and such that for each $1 \leq i \leq n$, A_1 (and hence A_2) contains exactly one of $\{a_{2i-1}, a_{2i}\}$?*

Restricted Even-Odd Partition. *Given a set $A = \{a_1, a_2, \ldots, a_{2n}\}$ of $2n$ positive integers, where $a_i < a_{i+1}$ for each $1 \leq i \leq 2n$ and $a_i > 4 \sum_{j=1}^{n} (a_{2j} - a_{2j-1})$ for each $1 \leq i \leq 2n$, is there a partition of A into two subsets A_1 and A_2 such that $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$ and such that for each $1 \leq i \leq n$, A_1 (and hence A_2) contains exactly one of $\{a_{2i-1}, a_{2i}\}$?*

Note that the Restricted Even-Odd Partition problem differs from the Even-Odd Partition problem in that it has one more constraint on the integers; i. e., $a_i > 4 \sum_{j=1}^{n} (a_{2j} - a_{2j-1})$ for each $1 \leq i \leq 2n$. The NP-completeness of the Restricted Even-Odd Partition problem can be established by a simple reduction from the Even-Odd Partition problem: see [1] for more examples of this technique. Let $A = \{a_1, a_2, \ldots, a_{2n}\}$ be an instance of the Even-Odd Partition problem and let $\delta(A) = \sum_{j=1}^{n} (a_{2j} - a_{2j-1})$. Let $b_1 \leq b_2 \leq \cdots \leq b_n$ be an arbitrary sequence of n positive integers. Consider the set $A' = \{a_1 + b_1, a_2 + b_1, \ldots, a_{2i-1} + b_i, a_{2i} + b_i, \ldots, a_{2n-1} + b_n, a_{2n} + b_n\}$ Clearly, A' has a solution if and only if A has a solution. Furthermore, $\delta(A') = \delta(A)$, where $\delta(A')$ is the value of the δ function defined above when applied to the integers in A'. Thus, we can increase the values of the integers in A without changing the solution of A and the value of the δ function. Now, if we let $b_i = 4\delta(A)$ for each $1 \leq i \leq n$, then the set A' becomes an instance of the Restricted Even-Odd Partition problem, and it has a solution if and only if A has a solution. Thus, we have the following lemma.

Lemma 1. *The Restricted Even-Odd Partition problem is NP-complete.*

We begin by showing how to construct a set of jobs from an instance of the Restricted Even-Odd Partition problem; an instance of the $NLJP(2)$ problem

will be obtained by adding one more job to the set, as we shall see later. Let $A = \{a_1, a_2, \ldots, a_{2n}\}$ be an instance of the Restricted Even-Odd Partition problem, and let $B = (\sum_{i=1}^{2n} a_i)/2$, $\delta_i = a_{2i} - a_{2i-1}$ for each $1 \leq i \leq n$, $\Delta = \max\{\delta_1, \delta_2, \ldots, \delta_n\}$ and $\delta = \sum_{i=1}^{n} \delta_i$. Since $a_i < a_{i+1}$ for each $1 \leq i \leq 2n$, we have $\delta_i > 0$ and hence $\delta_i < \delta$ for each $1 \leq i \leq n$. We construct a set JS' of $4n$ jobs from A, consisting of the following three sub-sets of jobs: $X = \{X_1, X_2, \ldots, X_n\}$, $Y = \{Y_1, Y_2, \ldots, Y_n\}$ and $Q = \{Q_1, Q_2, \ldots Q_{2n}\}$. We call the first group of n jobs the *X jobs*, the second group of n jobs the *Y jobs* and the third group of $2n$ jobs the *partition jobs*. The processing times, release times and due dates of the $4n$ jobs in JS' are defined as follows; see Figure 1 for the pattern of release times and due dates of the jobs in JS'.

$$\begin{cases} p(X_i) = a_{2i-1} - 2\delta_i + 2\Delta + 2\delta_{i+1} \\ r(X_i) = \sum_{j=1}^{i-1} a_{2j-1} + 2(i-1)\Delta + 2\delta_i \qquad 1 \leq i \leq n-1 \\ d(X_i) = r(X_i) + p(X_i) + 2\delta \end{cases}$$

$$\begin{cases} p(X_n) = a_{2n-1} - 2\delta_n + 2\Delta \\ r(X_n) = \sum_{j=1}^{n-1} a_{2j-1} + 2(n-1)\Delta + 2\delta_n \\ d(X_n) = r(X_n) + p(X_n) + 3\delta/2 \end{cases}$$

$$\begin{cases} p(Y_i) = 2\Delta \\ r(Y_i) = \sum_{j=1}^{i} a_{2j-1} + 2(i-1)\Delta \qquad 1 \leq i \leq n \\ d(Y_i) = r(Y_i) + p(Y_i) \end{cases}$$

$$\begin{cases} p(Q_{2i-1}) = a_{2i-1} \\ r(Q_{2i-1}) = r(X_i) \qquad 1 \leq i \leq n \\ d(Q_{2i-1}) = r(Q_{2i-1}) + p(Q_{2i-1}) \end{cases}$$

$$\begin{cases} p(Q_{2i}) = a_{2i} \\ r(Q_{2i}) = r(Q_{2i-1}) - 2\delta_i \qquad 1 \leq i \leq n \\ d(Q_{2i}) = r(Q_{2i}) + p(Q_{2i}) \end{cases}$$

A job J is said to be an *urgent job* if $d(J) = r(J) + p(J)$; otherwise it is called a *non-urgent job*. For an urgent job to be on-time, it has to be scheduled continuously from its release time to its due date without any delay. Observe that in JS', the *Y jobs* and the *partition jobs* are urgent jobs and the *X jobs* are non-urgent jobs. For each $1 \leq i \leq n$, we let $GRP(i) = \{X_i, Y_i, Q_{2i-1}, Q_{2i}\}$. In the next four lemmas, we will characterize an optimal schedule for JS' on two machines. First, we will characterize the jobs in $GRP(i)$ in the next lemma.

Lemma 2. *Let S be a schedule for JS' on two machines. For each $1 \leq i \leq n$, at most three out of the four jobs in $GRP(i)$ can be on-time in S. Furthermore,*

if exactly three jobs in $GRP(i)$ are on-time in S, then one of $\{Q_{2i}, Q_{2i-1}\}$ must be late.

Proof: As can be seen from Figure 1, only one of the jobs in $GRP(i)$ is available for processing in the time intervals $[r(Q_{2i}), r(Q_{2i-1})]$ and $[d(Y_i), d(X_i)]$. Thus, only one of the machines can be used to process the jobs in $GRP(i)$ in these two intervals. Hence, the total processing power that can be used to process the jobs in $GRP(i)$ is $r(Q_{2i-1}) - r(Q_{2i}) + 2(d(Y_i) - r(Q_{2i-1})) + d(X_i) - d(Y_i) = 2a_{2i-1} - 2\delta_i + 4\Delta + 2\delta + 2\delta_{i+1}$ ($= 2a_{2n-1} - 2\delta_n + 4\Delta + 3\delta/2$ if $i = n$). On the other hand, the total processing time of the jobs in $GRP(i)$ is $2a_{2i-1} + a_{2i} - 2\delta_i + 4\Delta + 2\delta_{i+1}$ ($= 2a_{2n-1} + a_{2n} + 4\Delta - 2\delta_n$ if $i = n$). Since $a_k > 4\delta$ for each $1 \leq k \leq 2n$, the total processing time of the jobs in $GRP(i)$ is larger than the total processing power, and hence it is impossible to have all four jobs in $GRP(i)$ to be on-time. If both Q_{2i-1} and Q_{2i} are on-time, then Y_i must be late since all three of them are urgent jobs. Furthermore, X_i must also be late since it cannot start until $d(Q_{2i})$ and $d(Q_{2i}) + p(X_i) > d(X_i)$. ∎

Lemma 3. *Let S be a schedule for JS' on two machines such that Q_1 is on-time in S. Then, there is a schedule S' such that Q_2 is on-time in S' and $NLJ(S) \geq NLJ(S')$.*

Proof: By Lemma 2, Q_2 can not be on-time in S and therefore both machines are idle in the time interval $[r(Q_2), r(Q_1)]$. Note that at the time $r(Q_1)$, the remaining processing time of Q_2 is less than the processing time of Q_1 if we were to start Q_2 at the time $r(Q_2)$. Hence, completing Q_2 on time, instead of Q_1, leads to a schedule no worse than S. ∎

By Lemma 3, we may assume that Q_1 is always late in any schedule for JS' on two machines.

Lemma 4. *If $Q_{2i-1}, 2 \leq i \leq n$, is on-time in a schedule S, then at least one machine is idle at any time instant in the time interval $[r(Q_{2i}), r(Q_{2i-1})]$ in S.*

Proof: Since Q_{2i-1} is on-time in S, Q_{2i} must be late in S, by Lemma 2. Therefore, only one job, namely X_{i-1}, can possibly be scheduled in the time interval $[r(Q_{2i}), r(Q_{2i-1})]$ in S. Since there are two machines, at least one machine is idle in the interval. ∎

Corollary 1. *If S is a schedule such that $Q_i' \in \{Q_{2i-1}, Q_{2i}\}$ is on-time in S for each $1 \leq i \leq n$ and $Q_1' = Q_2$, then the total idle time in the time interval $[r(Q_1), d(Y_n)]$ in S is at least $\delta + 2(B - \sum_{i=1}^{n} p(Q_i'))$.*

Proof: By Lemma 4, the total idle time in the time interval $[r(Q_1), d(Y_n)]$ in S is at least

$$l = \sum_{\substack{Q_i' = Q_{2i-1} \\ 1 \leq i \leq n}} 2\delta_i.$$

It is easy to see that l can be written as $l = 2\sum_{i=1}^{n}(p(Q_{2i}) - p(Q'_i))$. Since $\sum_{i=1}^{n} p(Q_{2i}) = B + \delta/2$, we have $l = \delta + 2(B - \sum_{i=1}^{n} p(Q'_i))$. ∎

Lemma 5. *Let S be a schedule for JS' on two machines such that $Q'_i \in \{Q_{2i-1}, Q_{2i}\}$ is on-time in S for each $1 \le i \le n$ and $Q'_1 = Q_2$. Then, we have $NLJ(S) = n$ if and only if $\sum_{i=1}^{n} p(Q'_i) \ge B$.*

Proof: If $\sum_{i=1}^{n} p(Q'_i) \ge B$, then we can construct a schedule S as follows: Schedule all the urgent jobs in $\{Y_1, Y_2, \ldots, Y_n\} \cup \{Q'_1, Q'_2, \ldots, Q'_n\}$ nonpreemptively as soon as they are released, and schedule all the X *jobs* in ascending order of their release times so that they can be completed as early as possible. Figure 2 shows how the jobs in $GRP(i)$ are scheduled in S. By a simple calculation, it is easy to verify that all the X *jobs* are completed no later than their due dates. Clearly, $NLJ(S) = n$.

To complete the proof, we will show that $\sum_{i=1}^{n} p(Q'_i) < B$ implies there is no schedule S with $NLJ(S) = n$. We prove this by contradiction, assuming there is a schedule S with $NLJ(S) = n$. By Lemma 2, in order to have $3n$ on-time jobs in S, X_i, Y_i and Q'_i in $GRP(i)$ must be on-time in S. Therefore, the total processing time of the $3n$ on-time jobs in S is $l_1 = 4n\Delta + \sum_{i=1}^{n} a_{2i-1} + \sum_{i=1}^{n} p(Q'_i) - 2\delta_1$. By Corollary 1, the total idle time in the time interval $[r(Q_1), d(Y_n)]$ in S is $l_2 = \delta + 2(B - \sum_{i=1}^{n} p(Q'_i))$. In addition, there are at least $2\delta_1$ and $3\delta/2$ idle times in the time intervals $[r(Q_2), r(Q_1)]$ and $[d(Y_n)d(X_n)]$, respectively, in S. Thus, in order to have $3n$ on-time jobs in S, the total processing power needed before the last due date $d(X_n)$ is at least $l_3 = l_1 + l_2 + 2\delta_1 + 3\delta/2 = 4n\Delta + \sum_{i=1}^{n} a_{2i-1} + \sum_{i=1}^{n} p(Q'_i) + 5\delta/2 + 2(B - \sum_{i=1}^{n} p(Q'_i))$. Since $B = \sum_{i=1}^{n} a_{2i-1} + \delta/2$, we have $l_3 = 4n\Delta + 2\sum_{i=1}^{n} a_{2i-1} + 3\delta + (B - \sum_{i=1}^{n} p(Q'_i))$. Since $l_3 > 2d(X_n) = 4n\Delta + 2\sum_{i=1}^{n} a_{2i-1} + 3\delta$, it is impossible to have $3n$ on-time jobs in S. Thus, $NLJ(S) > n$. ∎

Using the above lemmas, we can easily show that the $NLJP(2)$ problem is NP-complete.

Theorem 1. *The $NLJP(2)$ problem is NP-complete.*

Proof: The $NLJP(2)$ problem is clearly in NP since a nondeterministic Turing machine can guess a subset of on-time jobs and verify in polynomial time that the subset of jobs is feasible. To complete the proof, we reduce the Restricted Even-Odd Partition problem to the $NLJP(2)$ problem as follows. Let $JS = JS' \cup \{E\}$, where JS' is defined at the beginning of the section and E is a newly added job. We let $p(E) = 2\delta_1 + 5\delta/2$, $r(E) = 0$ and $d(E) = d(X_n)$. Finally, we choose K to be n. Clearly, the reduction can be done in polynomial time.

By Lemma 2, there can be at most $3n+1$ on-time jobs in any schedule S for JS on two machines, and hence $NLJ(S) \ge K$. By Lemma 5, if $\sum_{i=1}^{n} p(Q'_i) < B$ in a schedule S, where Q'_i, $1 \le i \le n$, is as defined in Lemma 5, then more than K jobs will be late in S. On the other hand, if $\sum_{i=1}^{n} p(Q'_i) > B$ in S, then by

Corollary 1, there are less than δ available processing time in the time interval $[r(Q_1), d(Y_n)]$ for the processing of E, and hence E will be late in S. Consequently, more than K jobs will be late in both cases. If $\sum_{i=1}^{n} p(Q_i') = B$ in S, then by Lemma 5, there are $3n$ on-time jobs in JS'. Furthermore, by Lemma 4 and Corollary 1, there is a total of δ available processing time in the time interval $[r(Q_1), d(Y_n)]$ for the processing of E. Since there are $2\delta_1$ and $3\delta/2$ available processing times in the time intervals $[r(Q_2), r(Q_1)]$ and $[d(Y_n), d(X_n)]$, respectively, in S, E can also be completed on time. Thus, $NLJ(S) \leq K$ if and only if $\sum_{i=1}^{n} p(Q_i') = B$, or equivalently, the instance of the Restricted Even-Odd Partition problem has a solution if and only if the constructed instance of the $NLJP(2)$ problem has a solution. ∎

Corollary 2. *The $NLJP(m)$ problem is NP-complete for every fixed $m \geq 2$.*

Proof: For $m \geq 3$, we simply create a set of $2n$ additional jobs $\{F_{j,1}, F_{j,2}, \ldots, F_{j,2n}\}$ for each additional machine, where $1 \leq j \leq m-2$. The processing times, release times and due dates of these new jobs are defined as follows.

$$\begin{cases} p(F_{j,2i-1}) &= (a_{2i-1} + 2\delta_i)/2, \\ r(F_{j,2i-1}) &= r(Q_{2i}), \\ d(F_{j,2i-1}) &= r(F_{j,2i-1}) + p(F_{j,2i-1}), \end{cases}$$

$$\begin{cases} p(F_{j,2i}) &= (a_{2i-1} - 2\delta_i)/2 + 2\Delta, \\ r(F_{j,2i}) &= d(F_{j,2i-1}), \\ d(F_{j,2i}) &= r(F_{j,2i}) + p(F_{j,2i}), \end{cases}$$

where $1 \leq i \leq n$ and $1 \leq j \leq m-2$.

Observe that the $2n(m-2)$ new jobs are all urgent jobs and $d(F_{j,2i}) = d(Y_i)$ for each $1 \leq i \leq n$ and $1 \leq j \leq m-2$. We now redefine $GRP(i)$ to be the set of $4 + 2(m-2)$ jobs $\{X_i, Y_i, Q_{2i-1}, Q_{2i}, F_{1,2i-1}, F_{1,2i}, F_{2,2i-1}, F_{2,2i}, \ldots, F_{m-2,2i-1}, F_{m-2,2i}\}$ for each $1 \leq i \leq n$. Figure 3 shows how the jobs in $GRP(i)$ are related to one another in terms of release times and due dates. Using the same argument as given before, we can show that there is a schedule with no more than n late jobs if and only if the instance A of the Restricted Even-Odd Partition problem has a solution. ∎

3. Concluding Remarks

In this paper we have shown that the problem of minimizing the number of late jobs on m identical machines is NP-hard for every fixed $m \geq 2$. To further delineate the boundary of the complexity of this problem, it will be interesting to answer the following question: Can the problem be shown to be solvable in pseudopolynomial time or shown to be strongly NP-hard on a fixed number of machines ? on an arbitrary number of machines ?

Acknowledgment

We wish to thank Gilbert H. Young for his helpful comments and suggestions.

References

1. Du, J. and Leung, J. Y-T, *Minimizing Total Tardiness on One Machine is NP-hard*, Math. Oper. Res. **15** (1990), 483–495.
2. Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA (1979).
3. Kise, H., Ibaraki, T and Mine, H., *A Solvable Case of the One-Machine Scheduling Problem with Ready and Due Times*, Oper. Res. **26** (1978), 121–126.
4. Lageweg, B. J., Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G., *Computer Aided Complexity Classification of Deterministic Scheduling Problems*, Report BW 138, Mathematisch Centrum, Amsterdam (1981).
5. Lawler, E. L., *Preemptive Scheduling of Uniform Parallel Machines to Minimize the Weighted Number of Late Jobs*, Report BW 105, Mathematisch Centrum, Amsterdam (1979).
6. Lawler, E. L., *Recent Results in the Theory of Machine Scheduling*, in "Mathematical Programming: The State of the Art", A. Bachem, M. Grotschel and B. Korte (eds.), Springer, 1982.
7. Lawler, E. L., *New and Improved Algorithms for Scheduling a Single Machine to Minimize the Weighted Number of Late Jobs*, Preprint, Computer Science Division, University of California, Berkeley (1989).
8. Lawler, E. L, Lenstra, J.K., and Rinnooy Kan, A.H.G., *Recent Development in Deterministic Sequencing and Scheduling: a Survey*, in "Deterministic and Stochastic Scheduling", M. A. H. Dempster et al. (eds.), D. Reidel Publishing Company, 1982, pp. 35–73.
9. Lawler, E. L. and Martel, C.U., *Scheduling Periodically Occurring Tasks on Multiple Processors*, Information Processing Letters **12(1)** (1981), 9–12.
10. Moore, J. M., *Sequencing n Jobs on One Machine to Minimize the Number of Tardy Jobs*, Management Sci. **15** (1968), 102–109.
11. Sidney, J. B., *An Extension of Moore's Due Date Algorithm*, in "Symp. on the Theory of Scheduling and its Applications", S. E. Elmaghraby (ed.), Springer, 1973, pp. 393–38.

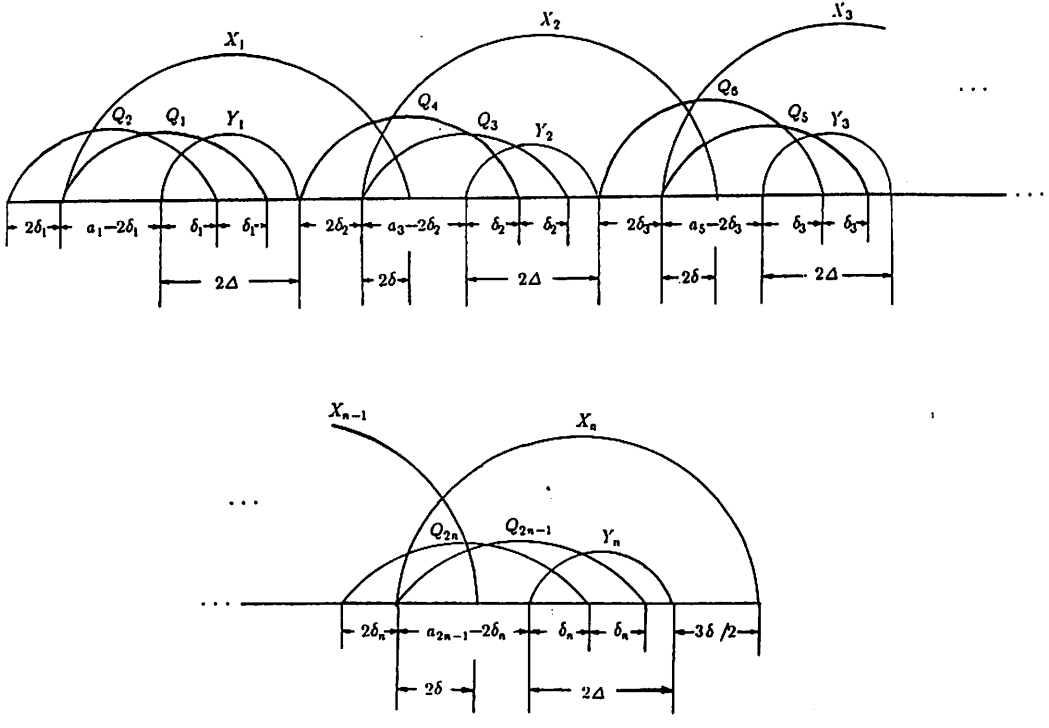

Figure 1. The Pattern of Release Times and Due Dates of the Jobs in JS'.

$$\delta_j' = \begin{cases} 2\delta_j & \text{If } Q_{2j-1} \text{ is on-time.} \\ \delta_j & \text{If } Q_{2j} \text{ is on-time.} \end{cases} \qquad \text{for } 1 \le j \le n$$

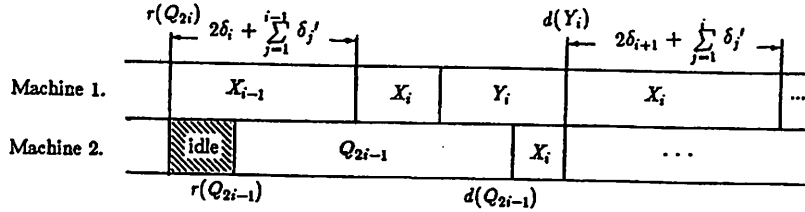

Figure 2(a). Q_{2i-1} is on-time.

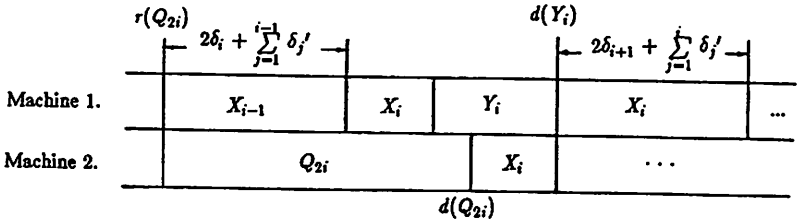

Figure 2(b). Q_{2i} is on-time.

Figure 2. Illustrating How to Schedule 3 Jobs in $GRP(i)$ on Time.

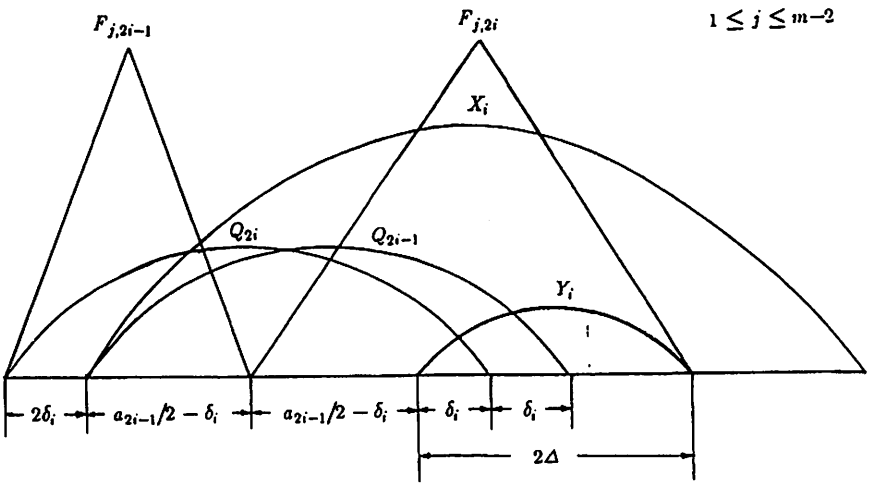

Figure 3. Release Times and Due Dates of the Jobs in $GRP(i)$ defined in Corollary 2.