MATHEMATICS OF OPERATIONS RESEARCH
Vol. 15, No. 3, August 1990
Printed in U.S.A.

MINIMIZING TOTAL TARDINESS ON ONE MACHINE IS *NP*-HARD*†

JIANZHONG DU AND JOSEPH Y.-T. LEUNG

University of Texas at Dallas

The problem of minimizing the total tardiness for a set of independent jobs on one machine is considered. Lawler has given a pseudo-polynomial-time algorithm to solve this problem. In spite of extensive research efforts for more than a decade, the question of whether it can be solved in polynomial time or it is *NP*-hard (in the ordinary sense) remained open. In this paper the problem is shown to be *NP*-hard (in the ordinary sense).

1. Introduction. We consider the problem of scheduling a set $\{J_1, J_2, \ldots, J_N\}$ of N independent jobs on one machine. Associated with each job J_i is a processing time $p(J_i)$ and a due date $d(J_i)$. If S is a schedule for the N independent jobs, then the *tardiness* of J_i in S, denoted by $T(J_i, S)$, is defined to be $T(J_i, S) = \max\{0, C(J_i, S) - d(J_i)\}$, where $C(J_i, S)$ denotes the completion time of J_i in S, and the *total tardiness* of S, denoted by $TT(S)$, is defined to be $TT(S) = \sum_{i=1}^{N} T(J_i, S)$. The problem is to find a schedule S for the N jobs on one machine such that the total tardiness $TT(S)$ is minimized. Such a schedule will be called an *optimal* schedule in this paper.

The total tardiness problem defined above was first studied by Emmons [4] in the late sixties. Up to the early seventies, all the work done in this area were basically practice oriented, aiming at designing fast enumerative algorithms to find an optimal schedule. Various solutions of this kind have been proposed; see [1], [4], [15] for examples. Typical results are concerned with the establishment of simple conditions, under which the next or the last job to be scheduled in an optimal schedule can easily be found. Consideration of complexity issues of this and related problems began in the late seventies. In [7] Lawler gives a pseudo-polynomial-time algorithm to find an optimal schedule; his algorithm also serves as a basis for a fully polynomial approximation scheme for this problem [9]. It is shown in [11] that if precedence constraints are introduced, then the problem becomes *NP*-hard even when all jobs have the same processing time.

The total tardiness problem is actually a special case of a more general scheduling problem introduced by McNaughton [13] in the late fifties. In the more general problem, each job J_i is additionally given a weight $w(J_i)$, and the objective is to find a schedule S for the N jobs on one or more parallel machines such that the *total weighted tardiness* of S, $\sum_{i=1}^{N} w(J_i)T(J_i, S)$, is minimized. In the case of one machine, McNaughton [13] has shown that preemption cannot reduce the total weighted tardiness for any given set of jobs. Thus, an optimal, preemptive schedule has the same total weighted tardiness as an optimal, nonpreemptive schedule. It has been

*Received March 16, 1988; revised March 2, 1989.
AMS 1980 subject classification. Primary: 90B35. Secondary: 68C25.
IAOR 1973 subject classification. Main: Combinatorial analysis.
OR/MS Index 1978 subject classification. Primary: 012 Analysis of algorithms/Computational complexity. Secondary: 581 Production/Scheduling.
Key words. *NP*-hard, one machine, independent job, due data, total tardiness.
†Research supported in part by the ONR grant N00014-87-K0833.

0364-765X/90/1503/0483/$01.25

shown that the problem of minimizing the total weighted tardiness on one machine is NP-hard in the strong sense [7], [12].

As pointed out in [2], [5], [6], [8], [10], [14], an important complexity issue for the total tardiness problem remained open over the years: Is the total tardiness problem solvable in polynomial time or is it NP-hard (in the ordinary sense)? In this paper we show that the problem is NP-hard (in the ordinary sense). Our result, together with Lawler's pseudo-polynomial-time algorithm, gives a sharp boundary on the complexity of this problem. Note that Lawler's pseudo-polynomial-time algorithm implies that the total tardiness problem cannot be NP-hard in the strong sense, unless $P = NP$.

A problem closely related to the total tardiness problem is the so-called total earliness problem. In the total earliness problem, we are also given N jobs, with each job having a processing time and a due date. However, the objective is to find a schedule S such that the *total earliness* of S, $TE(S) = \sum_{i=1}^{N} E(J_i, S)$, is minimized, where $E(J_i, S) = \max\{0, d(J_i) - C(J_i, S)\}$ is the *earliness* of J_i in S. The total tardiness problem and the total earliness problem are actually equivalent problems on one machine, in the sense that an algorithm for one problem can be used to solve the other problem. This follows from the observation that for each instance $J = \{J_1, J_2, \dots, J_N\}$ of the total tardiness problem, there is an instance $J' = \{J_1', J_2', \dots, J_N'\}$ of the total earliness problem such that the minimum total earliness of J' is the same as the minimum total tardiness of J, and conversely. Let $J = \{J_1, J_2, \dots, J_N\}$ be an instance of the total tardiness problem and let $C = \sum_{i=1}^{N} p(J_i)$. We construct an instance $J' = \{J_1', J_2', \dots, J_N'\}$ of the total earliness problem, where $p(J_i') = p(J_i)$ and $d(J_i') = C - d(J_i) + p(J_i)$ for each $1 \leqslant i \leqslant N$. Suppose S is an optimal schedule for J. We construct a schedule S' for J' as follows: If J_i is the kth job scheduled in S, then J_i' will be the $(N - k + 1)$th job scheduled in S'. Clearly, we have $C(J_i', S') = C - C(J_i, S) + p(J_i)$, and hence

$$\begin{aligned} E(J_i', S') &= \max\{0, d(J_i') - C(J_i', S')\} \\ &= \max\{0, C - d(J_i) + p(J_i) - (C - C(J_i, S) + p(J_i))\} \\ &= \max\{0, C(J_i, S) - d(J_i)\} = T(J_i, S). \end{aligned}$$

Thus, $TE(S') = TT(S)$. Similarly, if S' is an optimal schedule for J', then we can construct a schedule S for J such that $TT(S) = TE(S')$. Therefore, the minimum total earliness of J' is the same as the minimum total tardiness of J. Using the same technique as above, we can construct an instance J of the total tardiness problem from an instance J' of the total earliness problem such that the minimum total tardiness of J is the same as the minimum total earliness of J'. Thus, the two problems are equivalent on one machine, and hence they have the same complexity. Since we can show that the total tardiness problem is NP-hard, the total earliness problem must also be NP-hard.

In the next section we will state (without proof) some of the results given in [1], [4]; these results will be needed in the NP-hardness proof. Our reduction is from a restricted version of the NP-complete Even-Odd Partition problem [5]. We will show in §3 that the restricted Even-Odd Partition problem remains NP-complete. The NP-hardness proof of the total tardiness problem will be given in §4. Finally, we draw some conclusions in §5.

2. Preliminaries. In this section we state two lemmas that will be used in §4. Both lemmas follow directly from the results in [1], [4]. They are concerned with conditions under which some job should be, or should not be, scheduled next in an optimal schedule. Before we can state the lemmas, we need to introduce the following notations. Let S be a schedule of the N jobs $\{J_1, J_2, \ldots, J_N\}$ on one machine. For each $1 \leqslant k \leqslant N$, $F_k(S)$ denotes the kth job scheduled in S, and $t_k(S)$ denotes the completion time of the job $F_k(S)$ in S. The symbol $G_k(S)$ denotes the set of jobs scheduled after $F_k(S)$ in S. The next lemma follows directly from Theorem 1 in [4].

LEMMA 1. *Let S be an optimal schedule of the N jobs $\{J_1, J_2, \ldots, J_N\}$ on one machine. For each $1 \leqslant k \leqslant N-2$, if there is a job $J_i \in G_k(S)$ such that $p(J_i) \leqslant p(J_j)$ and $d(J_i) \leqslant \max\{t_k(S) + p(J_j), d(J_j)\}$ for all jobs $J_j \in G_k(S)$, then there is an optimal schedule S' such that $F_l(S') = F_l(S)$ for each $1 \leqslant l \leqslant k$ and $F_{k+1}(S')$ is the job J_i.*

Lemma 1 can be used to successively build up an optimal schedule. Suppose S is a partial schedule that can be extended to an optimal schedule. If there is an unscheduled job J_i satisfying the conditions of Lemma 1, then we can extend S by scheduling J_i next. In particular, if there is an unscheduled job having the smallest processing time and the earliest due date among all unscheduled jobs, then it should be scheduled next.

The next lemma can be used to decide which job should not be scheduled next. It is a generalization of the results described in Chapter 2 of [1]. The lemma can be proved by an interchange argument similar to that in [1].

LEMMA 2. *Let S be an optimal schedule of the N jobs $\{J_1, J_2, \ldots, J_N\}$ on one machine. For each $1 \leqslant k \leqslant N-2$, if there are two jobs J_i and J_j in $G_k(S)$ such that $p(J_i) > p(J_j)$ and $t_k(S) + p(J_i) > d(J_j)$, then there is an optimal schedule S' such that $F_l(S') = F_l(S)$ for each $1 \leqslant l \leqslant k$ and $F_{k+1}(S')$ is not the job J_i.*

Lemma 2 says that if there are two unscheduled jobs J_i and J_j, with J_i having a larger processing time than J_j, such that processing J_i next would pass the due date of J_j, then J_i should not be scheduled next. In particular, if all unscheduled jobs have already missed their due dates, then the remaining jobs should be scheduled in nondecreasing order of their processing times in an optimal schedule.

3. A restricted even-odd partition problem. In the next section we will show the *NP*-hardness of the total tardiness problem by a reduction from a restricted version of the *NP*-complete Even-Odd Partition problem [5]. In this section we establish the *NP*-completeness of this restricted Even-Odd Partition problem. Formally, the Even-Odd Partition problem and the Restricted Even-Odd Partition problem can be stated as follows.

Even-odd partition. Given a set of $2n$ positive integers $B = \{b_1, b_2, \ldots, b_{2n}\}$ such that $b_i > b_{i+1}$ for each $1 \leqslant i < 2n$, is there a partition of B into two subsets B_1 and B_2 such that $\sum_{b_i \in B_1} b_i = \sum_{b_i \in B_2} b_i$ and such that for each $1 \leqslant i \leqslant n$, B_1 (and hence B_2) contains exactly one of $\{b_{2i-1}, b_{2i}\}$?

Restricted even-odd partition. Given a set of $2n$ positive integers $A = \{a_1, a_2, \ldots, a_{2n}\}$ such that $a_i > a_{i+1}$ for each $1 \leqslant i < 2n$, $a_{2j} > a_{2j+1} + \delta$ for each $1 \leqslant j < n$, and $a_i > n(4n+1)\delta + 5n(a_1 - a_{2n})$ for each $1 \leqslant i \leqslant 2n$, where $\delta = \frac{1}{2}\sum_{i=1}^{n}(a_{2i-1} - a_{2i})$, 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 $1 \leqslant i \leqslant n$, A_1 (and hence A_2) contains exactly one of $\{a_{2i-1}, a_{2i}\}$?

LEMMA 3. *The Restricted Even-Odd Partition problem is NP-complete.*

PROOF. Clearly, the Restricted Even-Odd Partition problem, being a subproblem of the Even-Odd Partition problem, is in *NP*. We now give a reduction from the Even-Odd Partition problem to the Restricted Even-Odd Partition problem. Let $B = \{b_1, b_2, \ldots, b_{2n}\}$ be an arbitrary instance of the Even-Odd Partition problem. We construct an instance A of the Restricted Even-Odd Partition problem from B as follows.

$$a_1 = b_1 + (9n^2 + 3n)\Delta + 5n(b_1 - b_{2n}),$$

$$a_2 = b_2 + (9n^2 + 3n)\Delta + 5n(b_1 - b_{2n}),$$

$$\dots,$$

$$a_{2i-1} = b_{2i-1} + (9n^2 + 3n - i + 1)\Delta + 5n(b_1 - b_{2n}),$$

$$a_{2i} = b_{2i} + (9n^2 + 3n - i + 1)\Delta + 5n(b_1 - b_{2n}),$$

$$\dots,$$

$$a_{2n-1} = b_{2n-1} + (9n^2 + 2n + 1)\Delta + 5n(b_1 - b_{2n}),$$

$$a_{2n} = b_{2n} + (9n^2 + 2n + 1)\Delta + 5n(b_1 - b_{2n}),$$

where $\Delta = \frac{1}{2}\sum_{i=1}^{n}(b_{2i-1} - b_{2i})$. The construction can clearly be done in polynomial time.

Since $a_{2i-1} - a_{2i} = b_{2i-1} - b_{2i}$ for each $1 \leqslant i \leqslant n$, we immediately have $\delta = \Delta$. Since $b_i > b_{i+1}$ for each $1 \leqslant i < 2n$, it follows from the definition of a_i that $a_i > a_{i+1}$ for each $1 \leqslant i < 2n$. For each $1 \leqslant j < n$, $a_{2j} - a_{2j+1} = b_{2j} - b_{2j+1} + \Delta > \delta$. Thus, $a_{2j} > a_{2j+1} + \delta$ for each $1 \leqslant j < n$. Finally, since $a_1 - a_{2n} = b_1 - b_{2n} + (n-1)\Delta$ and since $\delta = \Delta$, it follows from the definition of a_i that $a_i > n(4n+1)\delta + 5n(a_1 - a_{2n})$ for each $1 \leqslant i \leqslant 2n$. Thus, the instance A satisfies the constraints of the Restricted Even-Odd Partition problem.

Suppose B_1 and B_2 constitute a solution to B. Then, we have $\sum_{b_i \in B_1} b_i = \sum_{b_i \in B_2} b_i$ and B_1 (hence, B_2) contains exactly one of $\{b_{2i-1}, b_{2i}\}$ for each $1 \leqslant i \leqslant n$. Let $A_1 = \{a_i \mid b_i \in B_1\}$ and $A_2 = \{a_i \mid b_i \in B_2\}$. Clearly, we have $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$ and A_1 (hence A_2) contains exactly one of $\{a_{2i-1}, a_{2i}\}$ for each $1 \leqslant i \leqslant n$. Hence, A_1 and A_2 constitute a solution to A. Conversely, if A_1 and A_2 constitute a solution to A, then we let $B_1 = \{b_i \mid a_i \in A_1\}$ and $B_2 = \{b_i \mid a_i \in A_2\}$. It is easy to see that $\sum_{b_i \in B_1} b_i = \sum_{b_i \in B_2} b_i$ and B_1 (hence B_2) contains exactly one of $\{b_{2i-1}, b_{2i}\}$ for each $1 \leqslant i \leqslant n$. Thus, B_1 and B_2 constitute a solution to B. Therefore, A has a solution if and only if B does. □

As we shall see in the next section, the reason for putting additional constraints on the integers in A is to facilitate our *NP*-hardness proof of the total tardiness problem. We note that, by using similar techniques as in Lemma 3, one can put different constraints on the integers in A and be able to show that the restricted version of the Even-Odd Partition problem is also *NP*-complete. We have exploited this fact in [3] to show the *NP*-hardness of another scheduling problem.

4. The total tardiness problem. In this section we show that the total tardiness problem is *NP*-hard by showing the corresponding decision problem to be *NP*-complete. The decision version of the total tardiness problem can be stated as follows.

Total tardiness. Given an integer ω and a set $\{J_1, J_2, \dots, J_N\}$ of N independent jobs, each job J_i having a processing time $p(J_i)$ and a due date $d(J_i)$, is there a schedule S of these N jobs on one machine such that $TT(S) \leqslant \omega$?

We begin by describing a reduction from the Restricted Even-Odd Partition problem to the Total Tardiness problem. Our reduction is similar in spirit to the one we give in [3], although a different problem is considered there. Let $A =$

$\{a_1, a_2, \ldots, a_{2n}\}$ be an arbitrary instance of the Restricted Even-Odd Partition problem and let $\bar{A} = \frac{1}{2}\sum_{i=1}^{2n} a_i$. Clearly, we have

$$\bar{A} = \sum_{i=1}^{n} a_{2i-1} - \delta = \sum_{i=1}^{n} a_{2i} + \delta.$$

Without loss of generality, we may assume that $a_{2i-1} \leqslant a_{2i} + \delta$ for each $1 \leqslant i \leqslant n$; for otherwise, there is no solution to A. We construct an instance of the Total Tardiness problem with $N = 3n + 1$ jobs $\{V_1, V_2, \ldots, V_{2n}\} \cup \{W_1, W_2, \ldots, W_{n+1}\}$. We call the first group of $2n$ jobs the V-jobs and the second group of $n + 1$ jobs the W-jobs. Let $b = (4n + 1)\delta$. The processing times and the due dates of the jobs are defined below; see Figure 1 for the pattern of the due dates of the N jobs.

$$p(V_i) = a_i \quad \text{for each } 1 \leqslant i \leqslant 2n.$$

$$p(W_i) = b \quad \text{for each } 1 \leqslant i \leqslant n + 1.$$

$$d(V_i) = \begin{cases} (j-1)b + \delta + (a_2 + a_4 + \cdots + a_{2j}) & \text{if } i = 2j - 1, \\ d(V_{2j-1}) + 2(n - j + 1)(a_{2j-1} - a_{2j}) & \text{if } i = 2j. \end{cases}$$

$$d(W_i) = \begin{cases} ib + (a_2 + a_4 + \cdots + a_{2i}) & \text{if } 1 \leqslant i \leqslant n, \\ d(W_n) + \delta + b & \text{if } i = n + 1. \end{cases}$$

Observe that the W-jobs have the same processing time and that each V-job corresponds to a partition element in A.

Let us consider a special kind of schedules for the above set of jobs. Suppose $\{V_{1,1}, V_{2,1}, \ldots, V_{n,1}\}$ and $\{V_{1,2}, V_{2,2}, \ldots, V_{n,2}\}$ is a partition of the V-jobs such that $\{V_{i,1}, V_{i,2}\} = \{V_{2i-1}, V_{2i}\}$ for each $1 \leqslant i \leqslant n$. We define a *canonical schedule* to be a schedule as shown in Figure 2. As can be seen in Figure 2, in a canonical schedule, the first group of n V-jobs, $V_{1,1}, V_{2,1}, \ldots, V_{n,1}$, and the first n W-jobs, $W_1, W_2, \ldots, W_n$, are scheduled alternately in the given order at the beginning, starting with $V_{1,1}$. Then the job W_{n+1} follows. Finally, the second group of n V-jobs, $V_{n,2}, V_{n-1,2}, \ldots, V_{1,2}$, are scheduled in the given order at the end. In the following we will establish that there is always an optimal schedule for the N jobs that is a canonical schedule. First, we need the following lemma.

LEMMA 4. *There is always an optimal schedule S_o for the N jobs such that either V_1 or V_2 is scheduled first in S_o.*

PROOF. Let S_o be an optimal schedule and let $\hat{V}$ be the first V-job scheduled in S_o. We want to show (1) $\hat{V}$ is the first job scheduled in S_o and (2) $\hat{V} \in \{V_1, V_2\}$. First, we show that $\hat{V}$ is the first job scheduled in S_o. Suppose that $\hat{V}$ is not the first job scheduled in S_o. Then, the jobs scheduled before $\hat{V}$ are all W-jobs. Since the W-jobs have the same processing times, we may assume that they are scheduled in increasing order of their due dates in S_o. Since $p(W_i) = b$ for each $1 \leqslant i \leqslant n + 1$ and since $a_i > (n + 1)b$ for each $1 \leqslant i \leqslant 2n$, none of the W-jobs scheduled before $\hat{V}$ can have their due dates missed in S_o. It is easy to verify that interchanging $\hat{V}$ with the W-job in front of $\hat{V}$ cannot increase the total tardiness of S_o. By a sequence of interchanges, we can move $\hat{V}$ to the front without increasing the total tardiness of S_o. Therefore, without loss of generality, we may assume that $\hat{V}$ is the first job scheduled in S_o.

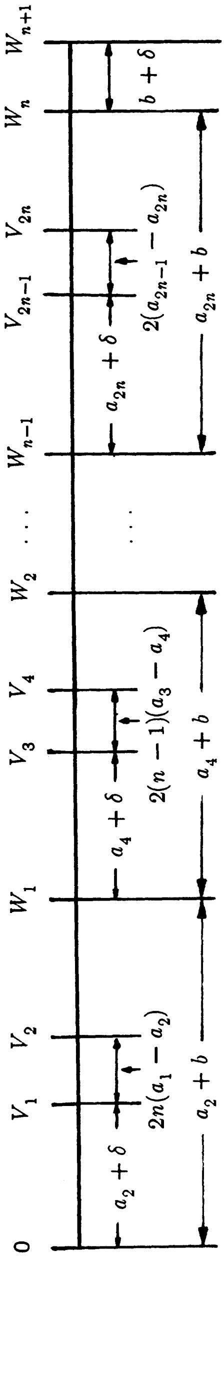

FIGURE 1. Due Date Pattern.

$V_{1,1}$	W_1	$V_{2,1}$	W_2	$\cdots$	W_{n-1}	$V_{n,1}$	W_n	W_{n+1}	$V_{n,2}$	$V_{n-1,2}$	$\cdots$	$V_{1,2}$

FIGURE 2. A Canonical Schedule.

To complete the proof, we need to show that $\hat{V} \in \{V_1, V_2\}$. Suppose $\hat{V}$ is not in $\{V_1, V_2\}$. Then, since $p(V_1) > p(V_2)$ and $p(\hat{V}) + p(V_1) > d(V_2)$, job V_2 must be scheduled before V_1 in S_o, by Lemma 2. Consider the schedule S' obtained from S_o by interchanging $\hat{V}$ with V_2. The tardiness of all jobs remain the same after the interchange, except possibly V_2, $\hat{V}$, and the jobs scheduled between $\hat{V}$ and V_2 in S_o. Since $T(\hat{V}, S_o) = 0$, $T(V_2, S_o) > 0$ and $T(V_2, S') = 0$, the tardiness of these jobs are related as follows.

$$T(V_2, S') = T(V_2, S_o) - (C(V_2, S_o) - d(V_2)),$$

$$T(\hat{V}, S') = T(\hat{V}, S_o) + \max\{0, C(V_2, S_o) - d(\hat{V})\}, \quad \text{and}$$

$$T(J, S') \leqslant T(J, S_o) + (p(V_2) - p(\hat{V}))$$

for each job J scheduled between $\hat{V}$ and V_2 in S_o. Since there are at most $3n - 2$ jobs scheduled between $\hat{V}$ and V_2, we have

$$TT(S') - TT(S_o) \leqslant (3n-2)(p(V_2) - p(\hat{V})) + \max\{0, C(V_2, S_o) - d(\hat{V})\}$$
$$-(C(V_2, S_o) - d(V_2)).$$

Now, if $C(V_2, S_o) - d(\hat{V}) \geqslant 0$, then we have

$$TT(S') - TT(S_o) \leqslant (3n-2)(p(V_2) - p(\hat{V})) - (d(\hat{V}) - d(V_2)).$$

Since $a_i > 5n(a_1 - a_{2n})$ for each $1 \leqslant i \leqslant 2n$ and since $b > 2n\delta \geqslant 2n(a_1 - a_2)$, we have $d(\hat{V}) - d(V_2) > 5n(a_1 - a_{2n})$. Since $p(V_2) - p(\hat{V}) \leqslant a_1 - a_{2n}$, we have

$$TT(S') - TT(S_o) \leqslant (3n-2)(a_1 - a_{2n}) - 5n(a_1 - a_{2n})$$
$$= -2(1+n)(a_1 - a_{2n}).$$

Thus, $TT(S') < TT(S_o)$, contradicting our assumption that S_o is an optimal schedule.

On the other hand, if $C(V_2, S_o) - d(\hat{V}) < 0$, then we have

$$\begin{aligned} TT(S') - TT(S_o) &\leqslant (3n-2)(p(V_2) - p(\hat{V})) - (C(V_2, S_o) - d(V_2)) \\ &\leqslant (3n-2)(p(V_2) - p(\hat{V})) - (p(\hat{V}) + p(V_2) - d(V_2)) \\ &= (3n-2)(p(V_2) - p(\hat{V})) - (p(\hat{V}) - \delta - 2n(a_1 - a_2)) \\ &= (3n-2)(p(V_2) - p(\hat{V})) + \delta + 2n(a_1 - a_2) - p(\hat{V}) \\ &\leqslant (5n-1)(a_1 - a_{2n}) - p(\hat{V}). \end{aligned}$$

Since $p(\hat{V}) > 5n(a_1 - a_{2n})$, we again have $TT(S') < TT(S_o)$, contradicting our assumption that S_o is an optimal schedule. Thus, $\hat{V}$ is in $\{V_1, V_2\}$. $\square$

LEMMA 5. *There is always an optimal schedule for the N jobs that is a canonical schedule.*

PROOF. Let S_o be an optimal schedule for the N jobs. We first show, by induction, that the V-jobs are divided into two groups, $\{V_{1,1}, V_{2,1}, \ldots, V_{n,1}\}$ and $\{V_{1,2}, V_{2,2}, \ldots, V_{n,2}\}$ ($\{V_{i,1}, V_{i,2}\} = \{V_{2i-1}, V_{2i}\}$ for each $1 \leqslant i \leqslant n$), such that the jobs $V_{1,1}, V_{2,1}, \ldots, V_{n,1}$ and the jobs $W_1, W_2, \ldots, W_n$ are scheduled in an alternate fashion at the beginning of S_o. As the basis case, we consider $i = 1$. By Lemma 4, we may assume that either V_1 or V_2 is scheduled first in S_o. Let the job that is scheduled first in S_o be called $V_{1,1}$, and the other job in $\{V_1, V_2\}$ be called $V_{1,2}$. After $V_{1,1}$ is completed, we may assume that W_1 is scheduled next in S_o, by Lemma 1.

Now suppose that for each $1 \leqslant i \leqslant k < n$, we have $\{V_{i,1}, V_{i,2}\} = \{V_{2i-1}, V_{2i}\}$, and $V_{1,1}, V_{2,1}, \ldots, V_{i,1}$ and $W_1, W_2, \ldots, W_i$ are scheduled in an alternate fashion at the beginning of S_o. We want to show that either V_{2k+1} or V_{2k+2} is scheduled next in S_o. Let t_k be the completion time of W_k in S_o. We have $t_k = \sum_{i=1}^{k}(p(V_{i,1}) + p(W_i))$. It is easy to see that $t_k \geqslant \sum_{i=1}^{k} a_{2i} + kb = d(W_k)$, and $t_k \leqslant \sum_{i=1}^{k} a_{2i} + 2\delta + kb = d(W_k) + 2\delta$. Since $p(V_{i,2}) > p(V_{2k+1}) + \delta > a_{2k+2} + \delta$ for each $1 \leqslant i \leqslant k$ and since $t_k \geqslant d(W_k)$, scheduling any job $V_{i,2}$ next will pass the due date of V_{2k+1}. Since $p(V_{i,2}) > p(V_{2k+1})$ for each $1 \leqslant i \leqslant k$, we may assume that none of the jobs $V_{i,2}$ is scheduled next in S_o, by Lemma 2. We now use the same argument as in the proof of Lemma 4 to show that either V_{2k+1} or V_{2k+2} is scheduled next in S_o. Let $\hat{V}$ be the first V-job scheduled after t_k in S_o. By the above argument, $\hat{V}$ is not in $\{V_{1,2}, V_{2,2}, \ldots, V_{k,2}\}$. By the same argument as in Lemma 4, we can show that $\hat{V}$ starts at t_k in S_o.

We now show that $\hat{V} \in \{V_{2k+1}, V_{2k+2}\}$. Suppose $\hat{V}$ is not in $\{V_{2k+1}, V_{2k+2}\}$. Then, since $p(V_{2k+1}) > p(V_{2k+2})$ and $t_k + p(\hat{V}) + p(V_{2k+1}) > d(V_{2k+2})$, job V_{2k+2} must be scheduled before V_{2k+1} in S_o, by Lemma 2. Note that by the above argument, the jobs scheduled between $\hat{V}$ and V_{2k+2} cannot be anyone of the jobs in $\{V_{1,2}, V_{2,2}, \ldots, V_{k,2}\}$. Consider the schedule S' obtained from S_o by interchanging $\hat{V}$ with V_{2k+2}. The tardiness of all jobs remain the same, except possibly for V_{2k+2}, $\hat{V}$, and the jobs scheduled between $\hat{V}$ and V_{2k+2} in S_o. Since $T(\hat{V}, S_o) = 0$, $T(V_{2k+2}, S_o) > 0$ and $t_k \leqslant d(W_k) + 2\delta$, the tardiness of these jobs are related as follows.

$$T(V_{2k+2}, S') \leqslant T(V_{2k+2}, S_o) - (C(V_{2k+2}, S_o) - d(V_{2k+2})) + \delta,$$

$$T(\hat{V}, S') = T(\hat{V}, S_o) + \max\{0, C(V_{2k+2}, S_o) - d(\hat{V})\}, \quad \text{and}$$

$$T(J, S') \leqslant T(J, S_o) + (p(V_{2k+2}) - p(\hat{V}))$$

for each job J scheduled between $\hat{V}$ and V_{2k+2} in S_o. Since there are at most $3n - 3k - 2$ jobs scheduled between $\hat{V}$ and V_{2k+2}, we have

$$TT(S') - TT(S_o) \leqslant (3n - 3k - 2)(p(V_{2k+2}) - p(\hat{V}))$$
$$+ \max\{0, C(V_{2k+2}, S_o) - d(\hat{V})\} - (C(V_{2k+2}, S_o) - d(V_{2k+2})) + \delta.$$

Now, if $C(V_{2k+2}, S_o) - d(\hat{V}) \geqslant 0$, then we have

$$TT(S') - TT(S_o) \leqslant (3n - 3k - 2)(p(V_{2k+2}) - p(\hat{V}))$$
$$-(d(\hat{V}) - d(V_{2k+2})) + \delta.$$

Since $a_i > 5n(a_1 - a_{2n})$ for each $1 \leqslant i \leqslant 2n$ and since $b > 2n\delta \geqslant 2n(a_1 - a_2)$, we have $d(\hat{V}) - d(V_{2k+2}) > 5n(a_1 - a_{2n})$. Since $p(V_{2k+2}) - p(\hat{V}) \leqslant a_1 - a_{2n}$ and $\delta \leqslant a_1 - a_{2n}$, we have

$$TT(S') - TT(S_o) \leqslant (3n - 3k - 2)(a_1 - a_{2n}) - 5n(a_1 - a_{2n}) + (a_1 - a_{2n})$$
$$= -(2n + 3k + 1)(a_1 - a_{2n}).$$

Thus, $TT(S') < TT(S_o)$, contradicting our assumption that S_o is an optimal schedule.

On the other hand, if $C(V_{2k+2}, S_o) - d(\hat{V}) < 0$, then we have

$$\begin{aligned}
&TT(S') - TT(S_o)\\
&\quad\leqslant (3n - 3k - 2)\big(p(V_{2k+2}) - p(\hat{V})\big) - \big(C(V_{2k+2}, S_o) - d(V_{2k+2})\big) + \delta\\
&\quad\leqslant (3n - 3k - 2)\big(p(V_{2k+2}) - p(\hat{V})\big)\\
&\qquad -\big(t_k + p(\hat{V}) + p(V_{2k+2}) - d(V_{2k+2})\big) + \delta\\
&\quad= (3n - 3k - 2)\big(p(V_{2k+2}) - p(\hat{V})\big) - p(\hat{V}) + \delta\\
&\qquad -\big(t_k + p(V_{2k+2}) - d(V_{2k+2})\big).
\end{aligned}$$

Since $t_k + p(V_{2k+2}) \geqslant d(W_k) + a_{2k+2} = d(V_{2k+2}) - 2(n - k)(a_{2k+1} - a_{2k+2}) - \delta$, we have

$$\begin{aligned}
TT(S') - TT(S_o) &\leqslant (3n - 3k - 2)\big(p(V_{2k+2}) - p(\hat{V})\big) - p(\hat{V}) + \delta\\
&\quad +\big(\delta + 2(n - k)(a_{2k+1} - a_{2k+2})\big)\\
&= (3n - 3k - 2)\big(p(V_{2k+2}) - p(\hat{V})\big) + 2\delta\\
&\quad + 2(n - k)(a_{2k+1} - a_{2k+2}) - p(\hat{V})\\
&\leqslant (5n - 5k)(a_1 - a_{2n}) - p(\hat{V}).
\end{aligned}$$

Since $p(\hat{V}) > 5n(a_1 - a_{2n})$, we again have $TT(S') < TT(S_o)$, contradicting our assumption that S_o is an optimal schedule. Thus, $\hat{V}$ is in $\{V_{2k+1}, V_{2k+2}\}$.

Let us denote $\hat{V}$ by $V_{k+1,1}$ and the other job in $\{V_{2k+1}, V_{2k+2}\}$ by $V_{k+1,2}$. After $V_{k+1,1}$ is completed, we may assume that the next job to be scheduled in S_o is W_{k+1}, by Lemma 1. Hence, by induction, we have proved that $\{V_{1,1}, V_{2,1}, \ldots, V_{n,1}\}$ and $\{W_1, W_2, \ldots, W_n\}$ are scheduled in an alternate fashion at the beginning of S_o.

After W_n is completed, we may assume that W_{n+1} is the next job to be scheduled in S_o, by Lemma 1. After W_{n+1} is completed, all of the remaining jobs have already missed their due dates. Therefore, by Lemma 2, they are scheduled in nondecreasing order of their processing times in S_o. Hence, S_o is a canonical schedule as shown in Figure 2. □

Lemma 5 enables us to focus only on canonical schedules in a search for an optimal schedule of the N jobs given at the beginning of the section. In the following we will establish a lower bound for the total tardiness of an arbitrary canonical schedule of the N jobs. To derive the lower bound, it will be more convenient to set up the following notations. Let S be an arbitrary canonical schedule of the N jobs. Let the V-jobs be partitioned into two groups $\{V_{1,1}, V_{2,1}, \ldots, V_{n,1}\}$ and $\{V_{1,2}, V_{2,2}, \ldots, V_{n,2}\}$

such that the first group of jobs is scheduled before the second group of jobs in S. Let $p(V_{i,1}) = v_{i,1}$ and $p(V_{i,2}) = v_{i,2}$ for each $1 \leqslant i \leqslant n$. We have $\{v_{i,1}, v_{i,2}\} = \{a_{2i-1}, a_{2i}\}$ for each $1 \leqslant i \leqslant n$. Let

$$\omega_o = \bar{A} + nC - n(n-1)b/2 - n\delta - \sum_{i=1}^{n} (n-i+1)(a_{2i-1} + a_{2i}),$$

where $C = (n+1)b + 2\bar{A}$ is the total processing time of all the jobs. The next lemma shows that ω_o is a lower bound for the total tardiness of any canonical schedule.

LEMMA 6. *If S is a canonical schedule, then $TT(S) \geqslant \omega_o$. Moreover, the equality holds if and only if $\sum_{i=1}^{n} v_{i,1} = \sum_{i=1}^{n} v_{i,2}$.*

PROOF. We first consider the total tardiness of the first n W-jobs in S, $\sum_{i=1}^{n} T(W_i, S)$. It is easy to verify that in S, W_i cannot be completed before its due date for each $1 \leqslant i \leqslant n$. Therefore, we have $C(W_i, S) \geqslant d(W_i)$ for each $1 \leqslant i \leqslant n$, and

$$\sum_{i=1}^{n} T(W_i, S) = \sum_{i=1}^{n} (C(W_i, S) - d(W_i)) = \sum_{i=1}^{n} C(W_i, S) - \sum_{i=1}^{n} d(W_i).$$

Since $C(W_i, S) = ib + \sum_{j=1}^{i} v_{j,1}$ for each $1 \leqslant i \leqslant n$, we obtain

$$\sum_{i=1}^{n} C(W_i, S) = n(n+1)b/2 + \sum_{i=1}^{n} (n-i+1) v_{i,1}.$$

In addition, we have

$$\sum_{i=1}^{n} d(W_i) = n(n+1)b/2 + \sum_{i=1}^{n} (n-i+1) a_{2i}.$$

Therefore, we have

$$\sum_{i=1}^{n} T(W_i, S) = \sum_{i=1}^{n} C(W_i, S) - \sum_{i=1}^{n} d(W_i)$$

$$= \sum_{i=1}^{n} (n-i+1)(v_{i,1} - a_{2i}).$$

We now consider the total tardiness of the V-jobs in $\{V_{1,2}, V_{2,2}, \ldots, V_{n,2}\}$. It is easy to see that $V_{i,2}$ always misses its due date in S. Therefore, $C(V_{i,2}, S) \geqslant d(V_{i,2})$ for each $1 \leqslant i \leqslant n$, and

$$\sum_{i=1}^{n} T(V_{i,2}, S) = \sum_{i=1}^{n} (C(V_{i,2}, S) - d(V_{i,2})) = \sum_{i=1}^{n} C(V_{i,2}, S) - \sum_{i=1}^{n} d(V_{i,2}).$$

From the observation that $C(V_{i,2}, S) = C - \sum_{j=1}^{i-1} v_{j,2}$ for each $1 \leqslant i \leqslant n$, we obtain

$$\sum_{i=1}^{n} C(V_{i,2}, S) = nC - \sum_{i=1}^{n-1} (n-i) v_{i,2}.$$

If $V_{i,2} = V_{2i-1}$, then $d(V_{i,2}) = (i-1)b + \delta + \sum_{j=1}^{i} a_{2j}$. On the other hand, if $V_{i,2} = V_{2i}$, then

$$d(V_{i,2}) = (i-1)b + \delta + \sum_{j=1}^{i} a_{2j} + 2(n-i+1)(a_{2i-1} - a_{2i}).$$

Therefore, we have

$$d(V_{i,2}) = (i-1)b + \delta + \sum_{j=1}^{i} a_{2j} + (n-i+1)(a_{2i-1} - a_{2i})$$
$$+(n-i+1)(v_{i,1} - v_{i,2}) \quad \text{for each } 1 \leqslant i \leqslant n,$$

and hence

$$\sum_{i=1}^{n} d(V_{i,2}) = n\delta + n(n-1)b/2 + \sum_{i=1}^{n} (n-i+1)a_{2i-1}$$
$$+ \sum_{i=1}^{n} (n-i+1)(v_{i,1} - v_{i,2}).$$

Therefore, we have

$$\sum_{i=1}^{n} T(V_{i,2}, S) = \sum_{i=1}^{n} C(V_{i,2}, S) - \sum_{i=1}^{n} d(V_{i,2})$$
$$= nC - n\delta - n(n-1)b/2 - \sum_{i=1}^{n} (n-i+1)a_{2i-1}$$
$$- \sum_{i=1}^{n} (n-i+1)v_{i,1} + \sum_{i=1}^{n} v_{i,2}.$$

Using the above formulas for the tardiness, we obtain

$$TT(S) = \sum_{i=1}^{n} T(W_i, S) + \sum_{i=1}^{n} T(V_{i,2}, S) + \sum_{i=1}^{n} T(V_{i,1}, S) + T(W_{n+1}, S)$$
$$= nC - n(n-1)b/2 - n\delta - \sum_{i=1}^{n} (n-i+1)(a_{2i-1} + a_{2i}) + \sum_{i=1}^{n} v_{i,2}$$
$$+ \sum_{i=1}^{n} T(V_{i,1}, S) + T(W_{n+1}, S).$$

If $\sum_{i=1}^{n} v_{i,1} \leqslant \bar{A}$, then it is easy to see that $\sum_{i=1}^{k} v_{i,1} \leqslant \sum_{i=1}^{k} a_{2i} + \delta$ for each $1 \leqslant k \leqslant n$. This implies that $T(V_{i,1}, S) = 0$ for each $1 \leqslant i \leqslant n$ and $T(W_{n+1}, S) = 0$. Therefore, $TT(S) \geqslant \omega_o$, and the equality holds if and only if $\sum_{i=1}^{n} v_{i,1} = \sum_{i=1}^{n} v_{i,2} = \bar{A}$. On the other hand, if $\sum_{i=1}^{n} v_{i,1} > \bar{A}$, then $T(W_{n+1}, S) = \sum_{i=1}^{n} v_{i,1} - \bar{A}$. In addition, there exist indexes j such that $\sum_{i=1}^{j} v_{i,1} > \sum_{i=1}^{j} a_{2i} + \delta$. Let k be the smallest such index. It is clear that $V_{k,1} = V_{2k-1}$ and that $V_{k,1}$ misses its due date in S. Thus, we have $T(V_{k,1}, S) > 0$. Substituting into the above formula, we see that $TT(S) > \omega_o$. Hence, the lemma is proved. □

Using Lemmas 5 and 6, we can prove that the Total Tardiness problem is *NP*-complete.

THEOREM 1. *The Total Tardiness problem is NP-complete.*

PROOF. The Total Tardiness problem is clearly in *NP*. For any given instance A of the Restricted Even-Odd Partition problem, we construct an instance of the Total Tardiness problem as described at the beginning of the section, and we let $\omega = \omega_o$. The construction can clearly be done in polynomial time. Suppose A_1 and A_2 constitute a solution to A. We partition the V-jobs into two groups $\{V_i \mid a_i \in A_1\}$ and $\{V_i \mid a_i \in A_2\}$. From this partition of the V-jobs, we construct a canonical schedule S_o. By Lemma 6, we have $TT(S_o) = \omega_o$. Thus, the constructed instance of the Total Tardiness problem has a solution. Conversely, if the constructed instance of the Total Tardiness problem has a solution, then there is an optimal schedule S_o such that $TT(S_o) \leqslant \omega_o$. By Lemma 5, we may assume that S_o is a canonical schedule. Let $A_1 = \{a_i \mid V_i$ is one of the first n V-jobs scheduled in $S_o\}$ and $A_2 = \{a_i \mid V_i$ is one of the last n V-jobs scheduled in $S_o\}$. By Lemma 6, we have $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$. Thus, A has a solution. □

5. Conclusions.

In this paper we have shown that the problem of minimizing the total tardiness on one machine is *NP*-hard (in the ordinary sense). Since McNaughton [13] has shown that preemption cannot reduce the total tardiness on one machine, the *NP*-hardness result applies to both preemptive and nonpreemptive scheduling disciplines. For parallel machines, it is easy to see that preemption *can* reduce the total tardiness. For future research, it will be interesting to investigate the complexity of this problem on parallel machines, both for preemptive and nonpreemptive scheduling disciplines. It is conceivable that some of these problems are *NP*-hard in the strong sense.

Acknowledgements. We would like to thank the anonymous referees for their suggestions in improving the readability of the paper.

References

[1] Baker, K. R. (1974). *Introduction to Sequencing and Scheduling*. Wiley, New York.
[2] Coffman, E. G., Jr. (Ed.) (1976). *Computer and Job-Shop Scheduling Theory*. Wiley, New York.
[3] Du, J. and Leung, J. Y-T. (1988). Minimizing Mean Flow Time with Release Time and Deadline Constraints. Technical Report UTDCS-2-88, Computer Science Program, University of Texas at Dallas, Richardson, TX 75083.
[4] Emmons, H. (1969). One-Machine Sequencing to Minimize Certain Functions of Job Tardiness. *Oper. Res.* **17** 701–715.
[5] Garey, M. R. and Johnson, D. S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.
[6] Lageweg, B. J., Lenstra, J. K., Lawler, E. L. and Rinnooy Kan, A. H. G. (1982). Computer-Aided Complexity Classification of Combinatorial Problems. *Comm. ACM* **25** 817–822.
[7] Lawler, E. L. (1977). A 'Pseudopolynomial' Algorithm for Sequencing Jobs to Minimize Total Tardiness. *Ann. Discrete Math.* **1** 331–342.
[8] ______. (1982). Recent Results in the Theory of Machine Scheduling. in A. Bachem, M. Grötschel and B. Korte (Eds.), *Mathematical Programming: The State of the Art*, Springer, Berlin and New York.
[9] ______. (1982). A Fully Polynomial Approximation Scheme for the Total Tardiness Problem. Technical Report, Computer Science Division, University of California at Berkeley, Berkeley.

[10] Lawler, E. L., Lenstra, J. K. and Rinnooy Kan, A. H. G. (1982). 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, Dordrecht, 35–73.
[11] Lenstra, J. K. and Rinnooy Kan, A. H. G. (1978). Complexity of Scheduling under Precedence Constraints. *Oper. Res.* **26** 22–35.
[12] ———, ——— and Brucker, P. (1977). Complexity of Machine Scheduling Problems. *Ann. Discrete Math.* **1** 343–362.
[13] McNaughton, R., (1959). Scheduling with Deadlines and Loss Functions. *Management Sci.* **6** 1–12.
[14] Rinnooy Kan, A. H. G. (1976). *Machine Scheduling Problems: Classification, Complexity and Computations*. Martinus Nijhoff, The Hague.
[15] ———, Lageweg, B. J. and Lenstra, J. K. (1975). Minimizing Total Costs in One-Machine Scheduling. *Oper. Res.* **23** 908–927.

COMPUTER SCIENCE PROGRAM, UNIVERSITY OF TEXAS AT DALLAS, RICHARDSON, TEXAS 75083