

# Complexity of Two Dual Criteria Scheduling Problems

Yumei Huo      Joseph Y-T. Leung      Hairong Zhao

Department of Computer Science  
New Jersey Institute of Technology  
Newark, NJ 07102, USA

yh23@njit.edu, leung@cis.njit.edu, hairong@cis.njit.edu

March 1, 2004

## Abstract

In this article we answer the complexity question of two dual criteria scheduling problems which had been open for a long time. Both problems are single machine scheduling problems and both have the number of tardy jobs as the primary criterion. However, the first problem has the total completion time and the second one has the total tardiness as the secondary criterion. We show that both problems are binary NP-hard.

## 1 Introduction

In the past, most of the research in scheduling has focussed on a single criterion. Numerous effective algorithms and heuristics have been developed for these single criterion problems; see Pinedo [14] and Brucker [1]. However, companies are usually faced with the problem of satisfying several different groups of people. Woolsey [18] described a problem faced by the scheduler at a southwestern company that needs to simultaneously satisfy both the customers and the salespeople. According to Panwalkar *et al.* [13], managers actually develop schedules based on multiple criteria. Unfortunately, schedules that are optimal for one criterion usually perform quite poorly for other criteria. Thus, there is a need

for further research in multi-criteria scheduling problems, and indeed, these problems have received more attention in the last three decades.

Consider a single machine scheduling problem, where  $n$  jobs are ready for processing at time 0. Each job  $k$  has a processing time  $p_k$  and a due date  $d_k$ . If  $S$  is a schedule of the  $n$  jobs, we let  $C_k$  denote the completion time of job  $k$  in  $S$ . If  $C_k > d_k$ , we say that job  $k$  is tardy and we let  $T_k = C_k - d_k$  denote its tardiness. In addition, we use the variable  $U_k$  as an indicator that job  $k$  is tardy; in this case  $U_k$  is set to 1. On the other hand, if  $C_k \leq d_k$ , we say that job  $k$  is on time, and we let  $U_k = 0$  and  $T_k = 0$ .

Most of the single criterion scheduling problems are concerned with minimizing the total completion time,  $\sum C_j$ ; the number of tardy jobs,  $\sum U_j$ ; the maximum tardiness,  $T_{\max} = \max\{T_j\}$ ; as well as the total tardiness,  $\sum T_j$ . Following the notation of Graham *et al.* [6], the above scheduling problems are denoted by  $1 \parallel \sum C_j$ ,  $1 \parallel \sum U_j$ ,  $1 \parallel T_{\max}$ , and  $1 \parallel \sum T_j$ , respectively.

It is well known that the SPT rule (shortest processing time first) gives a schedule with minimum total completion time. The SPT rule schedules jobs in ascending order of their processing times. Maximum tardiness can be minimized by the EDD rule (earliest due date first), which schedules jobs in ascending order of their due dates. A schedule with minimum number of tardy jobs can be obtained by the Hodgson-Moore algorithm [12], which schedules jobs in ascending order of due dates. In the course of scheduling, if there is a job, say  $k$ , that completes after its due date, then the longest job currently in the schedule (including job  $k$ ) will be deleted from the schedule. The deleted jobs will be scheduled after all the on-time jobs. While the above three problems are solvable in polynomial time, unfortunately, minimizing total tardiness is binary NP-hard, as shown by Du and Leung [5].

In the literature, dual criteria scheduling problems have been studied under three approaches. The first approach is to have one criterion designated as the primary criterion and the other one designated as the secondary criterion. Here, we seek a schedule that minimizes the primary criterion and choose, from among all the schedules that minimize the primary criterion, the one that also minimizes the secondary criterion. The second approach is to efficiently generate the Pareto curve which enables the decision maker to make explicit trade-offs between these schedules. The final approach is to minimize a cost function which is a linear combination of the two criteria. In this article we consider only the first approach. Although there are numerous work done under the second and the third approaches, we

shall not dwell into them in this article.

Extending the notation of Graham *et al.* [6], we use  $1 \parallel \gamma_2 \mid \gamma_1$  to denote the single machine scheduling problem, where  $\gamma_1$  is the primary criterion and  $\gamma_2$  is the secondary criterion. For example,  $1 \parallel \sum C_j \mid T_{\max}$  denotes the problem where the primary criterion is maximum tardiness and the secondary criterion is total completion time. As another example,  $1 \parallel \sum T_j \mid \sum U_j$  denotes the problem where the primary criterion is the number of tardy jobs and the secondary criterion is the total tardiness.

As early as 1956, Smith [16] developed a polynomial-time algorithm for the problem  $1 \parallel \sum C_j \mid T_{\max} = 0$ . Heck and Roberts [11] extended the algorithm to solve  $1 \parallel \sum C_j \mid T_{\max}$ , while Emmons [10] extended it to solve  $1 \parallel \sum C_j \mid \max\{f_j(C_j)\}$ , where  $f_j(C_j)$  is an arbitrary nondecreasing penalty function for job  $j$ .

In 1975, Emmons [9] studied the problem  $1 \parallel \sum C_j \mid \sum U_j$ . He proposed a branch-and-bound algorithm which in the worst case runs in exponential time. Complexity question was not addressed in [9]. Later, Chen and Bulfin [2] proved that the problem is NP-hard with respect to id-encoding. In id-encoding, jobs with the same characteristics are represented only once, and the number of jobs with the same characteristics is represented by a binary number. Notice that id-encoding scheme has the effect of significantly reducing the size of the input, making the problem harder to solve in polynomial time as a function of the size of the input. The complexity of the problem under standard encoding schemes (e.g., binary or unary) remained open until now. In this article, we show that this problem is binary NP-hard.

Vairaktarakis and Lee [17] studied the problem  $1 \parallel \sum T_j \mid \sum U_j$ . They gave a polynomial-time algorithm when the set of tardy jobs is specified. As well, a branch-and-bound algorithm was given for the general problem. Chen and Bulfin [2] mentioned that the complexity of this problem is open. In this article, we show that this problem is also binary NP-hard.

The problem  $1 \parallel \max\{T_j\} \mid \sum U_j$  has been studied by Shanthikumar [15]. He gave a polynomial-time algorithm when the set of tardy jobs is specified. A branch-and-bound algorithm was also given for the general problem. As remarked by Chen and Bulfin [2], the complexity of this problem is open. Interestingly, the complexity of  $1 \parallel \sum U_j \mid \max\{T_j\}$  is also open, see Chen and Bulfin [2]. In [3], Chen and Bulfin gave heuristic and branch-and-bound algorithms for this latter problem.

Further results about primary and secondary criteria scheduling problems can be found in Chen and Bulfin [2] and Dileepan and Sen [4].

Our NP-hardness proofs are obtained by reductions from the Even-Odd Partition problem, which is known to be binary NP-complete (see Garey and Johnson [7] and Garey *et al.* [8]).

**EVEN-ODD PARTITION** Given  $2n$  positive integers  $a_1 < a_2 < \dots < a_{2n}$  where  $A = \frac{1}{2} \sum_{j=1}^{2n} a_j$ , is there a partition of the integers into two sets,  $A_1$  and  $A_2$ , such that

$$\sum_{j \in A_1} a_j = \sum_{j \in A_2} a_j = A$$

where  $A_1$  and  $A_2$  each contains exactly one element of each pair  $\{a_{2i-1}, a_{2i}\}$ ,  $i = 1, 2, \dots, n$ ?

Notice that since each pair of integers,  $a_{2i-1}$  and  $a_{2i}$ , must be put into two different sets, we can add a constant  $c_i$  to each pair without changing the problem instance. By carefully choosing  $c_i$ , we may assume that the given instance of Even-Odd Partition satisfies the following properties:

$$a_1 > (2n + 2) \cdot \max_{1 \leq i \leq n} (a_{2i} - a_{2i-1}) \quad (1)$$

$$a_{2i-1} > \sum_{j=1}^{2i-2} a_j = \sum_{j=1}^{i-1} a_{2j} + \sum_{j=1}^{i-1} a_{2j-1} \quad (2)$$

The organization of this article is as follows. In Sections 2 and 3, we will give the NP-hardness proofs for  $1 \parallel \sum C_j \mid \sum U_j$  and  $1 \parallel \sum T_j \mid \sum U_j$ , respectively. In the last section, we will draw some concluding remarks.

## 2 Minimizing Total Completion Time Subject to Minimum $\sum U_j$

In this section, we show that  $1 \parallel \sum C_j \mid \sum U_j$  is binary NP-hard. We shall show that the decision version of this problem is binary NP-complete by reducing the Even-Odd Partition problem to it. Given an instance of the Even-Odd Partition problem,  $a_1 < a_2 < \dots < a_{2n}$ , where  $A = \frac{1}{2} \sum_{j=1}^{2n} a_j$ , we create an instance  $I$  of the scheduling problem as follows. There are  $2n$  P-jobs each of which

corresponds to an integer in the Even-Odd Partition instance,  $n$  *small* Q-jobs and a *large* R-job. The processing times and due dates of these jobs are shown in Table 1, where

$$x_i = a_{2i-1} - (2n - i + 2)(a_{2i} - a_{2i-1}) \quad (3)$$

for  $1 \leq i \leq n$ , and  $L$  is an integer greater than  $2A$ .

job	processing time	due date
$P_{2i-1}$	$a_{2i-1}$	$\sum_{k=1}^{i-1} a_{2k} + \sum_{k=1}^{i-1} x_k + a_{2i-1}$
$P_{2i}$	$a_{2i}$	$\sum_{k=1}^{i-1} a_{2k} + \sum_{k=1}^i x_k + a_{2i}$
$Q_i$	$x_i$	$\sum_{k=1}^{i-1} a_{2k} + \sum_{k=1}^i x_k + a_{2i-1}$
$R$	$L$	$\sum_{k=1}^n x_k + A + L$

Table 1: The processing times and due dates of the jobs in instance I.

Let the threshold for the total completion time be  $B$ , where

$$B = \sum_{j=1}^n a_{2j} \cdot (3n + 2 - 2j) + \sum_{j=1}^n a_{2j-1} \cdot (n - j + 1) + \sum_{j=1}^n x_j \cdot (3n + 3 - 2j) + (n + 1) L$$

$$+ \frac{1}{2} \sum_{j=1}^n (a_{2j} - a_{2j-1}) .$$

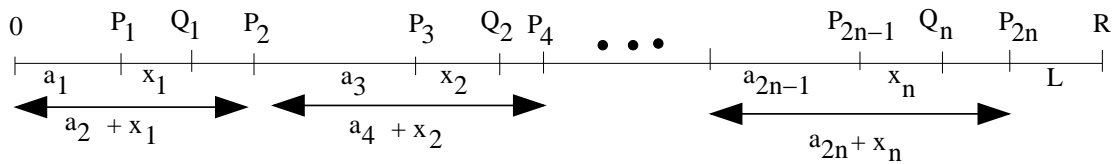


Figure 1: Illustration of the due dates of jobs in instance I.

Figure 1 shows the due date pattern of the jobs. Call a schedule *feasible* if it has the minimum number of tardy jobs. The decision problem asks: is there a feasible schedule with total completion time less than or equal to  $B$ ?

The basic idea of the reduction is to create a  $P$ -job for each integer  $a_j$ ,  $n$  small  $Q$ -jobs each of which has a due date between a pair of  $P$ -jobs, and a large  $R$ -job whose due date is the largest among

all the jobs. By properly choosing the processing times and due dates of the jobs, we can show that in any feasible schedule: (a) Exactly one job from each pair  $\{P_{2i-1}, P_{2i}\}$  must be tardy; (b) The Q-jobs must be on time; (c) The R-job must be on time and is scheduled after all the other on-time jobs and before any tardy jobs; (d) The total processing time of the on-time P-jobs cannot exceed  $A$ . However, to minimize total completion time, we need to have more even P-jobs to be on time. It can be shown that every time we interchange a pair of even and odd P-jobs by making the even P-job tardy and the odd P-job on time, the total completion time is increased by a quantity equal to the difference between the processing times of the two jobs, which is exactly the quantity reduced in the total processing time of the on-time P-jobs. Thus, the optimal solution is obtained when the total processing time of the on-time P-jobs is exactly  $A$ . But this occurs only when there is a solution to the instance of the Even-Odd Partition problem. Notice that the first three terms in the formula for  $B$  represent the total completion time when all even P-jobs are on time (which does not yield a feasible schedule since the R-job will be tardy). The last term is the minimum increase in total completion time when the total processing time of the on-time P-jobs is reduced to  $A$  (which yields a feasible schedule since the R-job will then be on time).

The next three lemmas prove the assertions made above.

**Lemma 2.1** *In any feasible schedule for the instance  $I$ , (a) there are exactly  $n$  tardy jobs, (b) the R-job is on time and is scheduled after all the other on-time jobs, and (c) at least one job from each pair  $\{P_{2i-1}, P_{2i}\}$  must be on time.*

**Proof :** We first prove (a), i.e., there must be  $n$  tardy jobs in any feasible schedule. As mentioned in the Introduction, the Hodgson-Moore algorithm yields a schedule with the minimum number of tardy jobs. Thus, it is sufficient to show that there are exactly  $n$  tardy jobs when the Hodgson-Moore algorithm is applied to the instance  $I$ .

Recall that Hodgson-Moore algorithm schedules jobs in increasing order of their due dates. In the course of scheduling, if a job misses its due date, then the job with the largest processing time among all jobs that are currently in the schedule (including the job that misses its due date), will be picked out as a tardy job and deleted from the schedule. We continue to schedule the next job until all jobs have been processed. Finally, we schedule all the tardy jobs (that were deleted from the schedule) at the end, in any order.

For the instance  $I$ , we have  $d_{P_{2i-2}} < d_{P_{2i-1}} < d_{Q_i} < d_{P_{2i}}$  and  $d_{P_{2n}} < d_R$ . Therefore, the jobs would be

scheduled in the order of  $P_1, Q_1, P_2, P_3, Q_2, P_4, \dots, P_{2n-1}, Q_n, P_{2n}, R$  by the Hodgson-Moore algorithm. It is easy to see that  $P_1$  and  $Q_1$  both meet their due dates, but  $P_2$  will not. By (1) and (3),  $x_1 < a_1 < a_2$ . Hence,  $P_2$  will be chosen as a tardy job. Suppose we have scheduled all the jobs  $P_{2j-1}, Q_j$  and  $P_{2j}$ , where  $1 \leq j \leq i-1$ , and all the even P-jobs,  $P_{2j}, 1 \leq j \leq i-1$ , had been chosen as tardy jobs and discarded from the schedule. If we now schedule  $P_{2i-1}, Q_i$  and  $P_{2i}$  sequentially, then the completion times will be

$$C_{P_{2i-1}} = \sum_{j=1}^{i-1} a_{2j-1} + \sum_{j=1}^{i-1} x_j + a_{2i-1} < \sum_{j=1}^{i-1} a_{2j} + \sum_{j=1}^{i-1} x_j + a_{2i-1} = d_{P_{2i-1}},$$

$$C_{Q_i} = \sum_{j=1}^{i-1} a_{2j-1} + \sum_{j=1}^i x_j + a_{2i-1} < \sum_{j=1}^{i-1} a_{2j} + \sum_{j=1}^i x_j + a_{2i-1} = d_{Q_i},$$

and

$$\begin{aligned} C_{P_{2i}} &= \sum_{j=1}^{i-1} a_{2j-1} + \sum_{j=1}^i x_j + a_{2i-1} + a_{2i} \\ &> \sum_{j=1}^i x_j + a_{2i-1} + a_{2i} \\ &> \sum_{j=1}^i x_j + \sum_{j=1}^{i-1} a_{2j} + a_{2i} \quad \text{by (2)} \\ &= d_{P_{2i}}. \end{aligned}$$

Since  $P_{2i}$  misses its due date and it has the largest processing time among all the jobs currently in the schedule,  $P_{2i}$  will be chosen as a tardy job. Therefore, the Hodgson-Moore algorithm will pick all the even P-jobs as tardy jobs. For the R-job, the completion time will be

$$\sum_{j=1}^n x_j + \sum_{j=1}^n a_{2j-1} + L < \sum_{j=1}^n x_j + A + L = d_R,$$

so it is on time. Hence, the total number of tardy jobs is  $n$ . Thus, any feasible schedule for the instance I must have exactly  $n$  tardy jobs.

Since the R-job has a large processing time, a job scheduled after the R-job must miss its due date. Hence, all the other on-time jobs must be scheduled before the R-job. Thus, (b) also holds.

We use contradiction to prove (c): at least one job from each pair  $\{P_{2i-1}, P_{2i}\}$  is on time. Suppose  $\{P_{2i-1}, P_{2i}\}$

is the first pair that are both tardy in a feasible schedule  $S$ . Consider now applying the Hodgson-Moore algorithm to the job set consisting of all Q-jobs, all P-jobs except  $\{P_{2i-1}, P_{2i}\}$ , and the R-job. As shown in the proof of (a), all the even P-jobs,  $P_{2j}$ ,  $j < i$ , will be picked as tardy jobs by the Hodgson-Moore algorithm. If we schedule  $\{P_{2i+1}, Q_{i+1}, P_{2i+2}\}$ , then  $P_{2i+1}$  and  $Q_{i+1}$  will still be on time. However, the completion time of  $P_{2i+2}$  is

$$\sum_{j=1}^{i-1} a_{2j-1} + \sum_{j=1}^{i+1} x_j + a_{2i+1} + a_{2i+2} > \sum_{j=1}^{i+1} x_j + a_{2i+1} + a_{2i+2} .$$

By (2),  $a_{2i+1} > \sum_{j=1}^i a_{2j}$ . Thus,  $P_{2i+2}$  will miss its due date. Since  $P_{2i+2}$  has the largest processing time among all jobs currently in the schedule, it will be chosen as a tardy job. Using the same argument, we can show that all even P-jobs are tardy. By assumption,  $P_{2i-1}$  is also a tardy job. Thus, the total number of tardy jobs will be  $n + 1$ , contradicting our assumption that  $S$  is a feasible schedule.  $\square$

**Lemma 2.2** *In any optimal schedule, exactly one job from each pair  $\{P_{2i-1}, P_{2i}\}$ ,  $1 \leq i \leq n$ , is tardy.*

**Proof :** We shall prove by contradiction that one of  $P_{2i-1}$  and  $P_{2i}$  must be tardy in any optimal schedule. Suppose both are on time in an optimal schedule  $S$ . By the proof in Lemma 2.1, one job in  $\{P_{2i-1}, Q_i, P_{2i}\}$  must be tardy. Since  $P_{2i-1}$  and  $P_{2i}$  are both on time,  $Q_i$  must be tardy. Consider now interchanging  $Q_i$  with  $P_{2i-1}$  (i.e., make  $P_{2i-1}$  tardy and  $Q_i$  on time) to get a new schedule  $S'$ . By (1) and (3), we have  $x_i < a_{2i-1}$ . On the other hand,  $d_{Q_i} > d_{P_{2i-1}}$ . So,  $Q_i$  meets its due date in  $S'$ . However,  $S'$  has a smaller total completion time than  $S$ , contradicting our assumption that  $S$  is optimal.  $\square$

**Lemma 2.3** *In any optimal schedule  $S$ , no tardy job can be scheduled before the R-job.*

**Proof :** By Lemma 2.1, we know that the on-time P-jobs and Q-jobs are all scheduled before the R-job. The total processing time of these jobs is at least  $\sum_{j=1}^n a_{2j-1} + \sum_{j=1}^n x_j$ . Suppose there is a tardy job scheduled before the R-job. By Lemma 2.2, we know that it must be a P-job. Suppose this job is  $P_m$ , where  $1 \leq m \leq 2n$ . Then the completion time of the R-job would be

$$\begin{aligned} & \sum_{j=1}^n a_{2j-1} + \sum_{j=1}^n x_j + L + a_m \\ & > \sum_{j=1}^n a_{2j-1} + \sum_{j=1}^n x_j + L + a_1 \\ & > \sum_{j=1}^n a_{2j-1} + \sum_{j=1}^n x_j + L + \sum_{j=1}^n (a_{2j} - a_{2j-1}) \quad \text{by (1)} \end{aligned}$$

$$\begin{aligned}
&= \sum_{j=1}^n a_{2j} + \sum_{j=1}^n x_j + L \\
&> A + \sum_{j=1}^n x_j + L .
\end{aligned}$$

Thus, the R-job will miss its due date. By Lemma 2.1,  $S$  can not be a feasible schedule.  $\square$

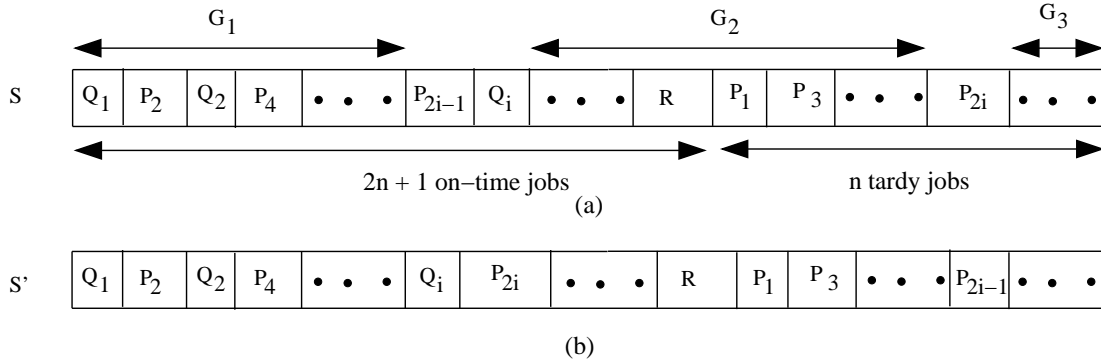


Figure 2: (a) A feasible schedule of jobs in the instance  $I$ , (b) The schedule obtained from (a) by interchanging  $P_{2i}$  with  $P_{2i-1}$ .

According to Lemmas 2.1 and 2.2, in any optimal schedule, all the on-time jobs must be scheduled before any tardy jobs. One can easily show that the on-time jobs must be scheduled in increasing order of their due dates in order to be on time. For the tardy jobs, it can be shown (by interchange argument) that in order to minimize the total completion time, they must be scheduled in increasing order of their processing times, which is the same order as the due dates of the  $P$ -jobs. There are only two possible configurations for each triplet  $\{P_{2i-1}, Q_i, P_{2i}\}$ , see Figure 2(a). We either have  $P_{2i-1}$  and  $Q_i$  on time and scheduled in this order, or we have  $Q_i$  and  $P_{2i}$  on time and scheduled in this order.

We now show that in order to minimize the total completion time, it is always better to pick  $Q_i$  and  $P_{2i}$  to be on time. Suppose there is a feasible schedule in which  $P_{2i-1}$  and  $Q_i$  are on time and  $P_{2i}$  is tardy. We shall show that by changing the configuration to  $Q_i$  and  $P_{2i}$  on time (see Figure 2(b)), the total completion time will be decreased by exactly  $a_{2i} - a_{2i-1}$ . We denote the original schedule as  $S$ , and the new schedule as  $S'$ . We use  $G_1$  to denote the jobs scheduled before  $P_{2i-1}$  in  $S$ ,  $G_2$  to denote the jobs scheduled between  $Q_i$  and  $P_{2i}$ , and  $G_3$  to denote the jobs scheduled after  $P_{2i}$  in  $S$ . It is easy to see that there are  $2n - i$  jobs in  $G_2$ .

Note that for each job in  $G_1$  and  $G_3$ , its completion time in  $S'$  remains the same as in  $S$ . For each job in  $G_2$ , the completion time will increase by  $a_{2i} - a_{2i-1}$ . So the total increase is  $(2n - i)(a_{2i} - a_{2i-1})$ . The completion time of  $Q_i$  decreases by  $a_{2i-1}$ . The completion time of  $P_{2i-1}$  in  $S'$  is the same as the completion time of  $P_{2i}$  in  $S$ . The completion time of  $P_{2i}$  in  $S'$  is larger than the completion time of  $P_{2i-1}$  by  $x_i + (a_{2i} - a_{2i-1})$ . Thus, the total completion time of all jobs is decreased by:

$$\begin{aligned}
& a_{2i-1} - (x_i + (a_{2i} - a_{2i-1})) - (2n - i)(a_{2i} - a_{2i-1}) \\
&= a_{2i-1} - x_i - (2n - i + 1)(a_{2i} - a_{2i-1}) \\
&= (2n - i + 2)(a_{2i} - a_{2i-1}) - (2n - i + 1)(a_{2i} - a_{2i-1}) \quad \text{by (3)} \\
&= (a_{2i} - a_{2i-1}) .
\end{aligned}$$

On the other hand, in order to ensure that the  $R$ -job is on time, we can not pick all the even  $P$ -jobs to be on time. Suppose we pick all the even  $P$ -jobs to be on time. Then the completion time of the  $R$ -job will be

$$\sum_{j=1}^n a_{2j} + \sum_{j=1}^n x_j + L > A + \sum_{j=1}^n x_j + L = d_R .$$

Therefore, the  $R$ -job will miss its due date. So, we must choose some even  $P$ -jobs as tardy jobs. Let  $B'$  be the total completion time when all the even  $P$ -jobs are on time. Then, we have  $B = B' + \frac{1}{2} \sum_{j=1}^n (a_{2j} - a_{2j-1})$ . As we have shown above, each time we interchange  $P_{2i}$  with  $P_{2i-1}$ , the total completion time will increase by  $a_{2i} - a_{2i-1}$ . At the same time, the completion time of the  $R$ -job will decrease by exactly  $a_{2i} - a_{2i-1}$ . So, if we can schedule the jobs such that the  $R$ -job completes at exactly its due date, then the total completion time has the minimum value  $B$  among all feasible schedules, and the total processing time of all the on-time  $P$ -jobs is exactly  $A$ . This means that there is a solution to the instance of the Even-Odd Partition problem if and only if there is a solution to the instance of the scheduling problem.

From the above discussions, we have the following theorem.

**Theorem 1** *The problem  $1 \parallel \sum C_j \mid \sum U_j$  is binary NP-hard.*

### 3 Minimizing Total Tardiness Subject to Minimum $\sum U_j$

In this section, we show that the Even-Odd Partition problem can be reduced to the decision version of the  $1 \parallel \sum T_j \mid \sum U_j$  problem. Given an instance of the Even-Odd Partition problem, we create an instance II of the scheduling problem as follows. There are  $2n$  P-jobs and a R-job. The processing times and due dates of these jobs are shown in Table 2, where  $L$  is an integer greater than  $A$ .

job	processing time	due date
$P_{2i-1}$	$a_{2i-1}$	$\sum_{k=1}^i a_{2k} + i \cdot (a_{2i} - a_{2i-1})$
$P_{2i}$	$a_{2i}$	$\sum_{k=1}^i a_{2k}$
R	$L$	$A + L$

Table 2: The processing times and due dates of the jobs in the instance II.

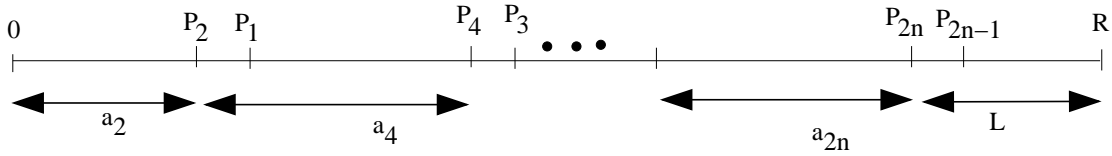


Figure 3: Illustration of the due dates of jobs in the instance II.

The due date pattern of the jobs are shown in Figure 3. Let the threshold for the total tardiness be  $B$ , where

$$B = \sum_{i=1}^n \left( L + \sum_{k=i+1}^n a_{2k} + \sum_{k=1}^i a_{2k-1} - i \cdot (a_{2i} - a_{2i-1}) \right) + \frac{1}{2} \cdot \sum_{i=1}^n (a_{2i} - a_{2i-1}) .$$

The decision problem asks: is there a schedule with the minimum number of tardy jobs such that the total tardiness is less than or equal to  $B$ ?

The basic idea of the reduction is similar to that in Section 2. For every pair of even and odd P-jobs, one of them must be on time and the other must be tardy. The R-job must be on time. To minimize total tardiness, it will be more advantageous to schedule all even P-jobs to be on time and all odd P-jobs to be tardy. But if all even P-jobs are on time, then the R-job will miss its due date by  $A$ . To ensure that the R-job is on time, the total processing time of all the on-time P-jobs cannot exceed

A. The total tardiness will attain its minimum, B, when the total processing time of all the on-time P-jobs is exactly A. Thus, there is a solution to the instance of the Even-Odd Partition problem if and only if there is a solution to the instance of the scheduling problem.

The next lemma proves the assertions made above.

**Lemma 3.1** *In any feasible schedule for the scheduling problem instance II, (a) there are exactly  $n$  tardy jobs, (b) the R-job is on time and all the on-time P-jobs are scheduled before the R-job, (c) exactly one job from each pair  $\{P_{2i-1}, P_{2i}\}$ ,  $1 \leq i \leq n$ , is tardy, (d) all tardy jobs are scheduled after the R-job.*

**Proof :** We prove (a) by showing that there are  $n$  tardy jobs when the Hodgson-Moore algorithm is applied to the instance II. For the instance II, we have  $d_{P_{2i}} < d_{P_{2i-1}} < d_{P_{2i+2}}$  and  $d_{P_{2n-1}} < d_R$ . Therefore, the jobs would be scheduled by the Hodgson-Moore algorithm in the order of  $P_2, P_1, P_4, P_3, \dots, P_{2n}, P_{2n-1}, R$ . It is easy to see that  $P_2$  meets its due date, but  $P_1$  will not. Since  $a_1 < a_2$ ,  $P_2$  will be chosen as a tardy job by the Hodgson-Moore algorithm. Suppose we have scheduled all the jobs  $P_{2j}$  and  $P_{2j-1}$ , where  $1 \leq j \leq i-1$ , and all the even P-jobs,  $P_{2j}$ ,  $1 \leq j \leq i-1$ , were chosen as tardy jobs. Consider now the scheduling of  $P_{2i}$  and  $P_{2i-1}$ . The completion times will be

$$C_{P_{2i}} = \sum_{j=1}^{i-1} a_{2j-1} + a_{2i} < \sum_{j=1}^{i-1} a_{2j} + a_{2i} = d_{P_{2i}} ,$$

and

$$\begin{aligned} C_{P_{2i-1}} &= \sum_{j=1}^{i-1} a_{2j-1} + a_{2i-1} + a_{2i} \\ &> \sum_{j=1}^{i-1} a_{2j-1} + \left( \sum_{j=1}^{i-1} (a_{2j-1} + a_{2j}) \right) + a_{2i} && \text{by (2)} \\ &= 2 \sum_{j=1}^{i-1} a_{2j-1} + \sum_{j=1}^i a_{2j} \\ &> 2(i-1) \cdot a_1 + \sum_{j=1}^i a_{2j} \\ &> n \cdot \max_{1 \leq j \leq n} (a_{2j} - a_{2j-1}) + \sum_{j=1}^i a_{2j} && \text{by (1)} \\ &> d_{P_{2i-1}} . \end{aligned}$$

Since  $P_{2i-1}$  misses its due date and since  $P_{2i}$  has the largest processing time among all jobs in the current schedule,  $P_{2i}$  will be chosen as a tardy job. Therefore, the Hodgson-Moore algorithm will pick all the even P-jobs as tardy jobs. For the R-job, the completion time will be

$$\sum_{j=1}^n a_{2j-1} + L < A + L = d_R ,$$

so it is on time. Hence the total number of tardy jobs is  $n$ , concluding the proof of (a).

Since the R-job has a large processing time, a job scheduled after the R-job must miss its due date. Hence, all the other on-time jobs must be scheduled before the R-job. This proves (b).

By (a) and (b), all tardy jobs are P-jobs. In order to prove (c), it is sufficient to prove that at least one job from each pair  $\{P_{2i-1}, P_{2i}\}$  must be on time. We shall prove this by contradiction. Suppose  $\{P_{2i-1}, P_{2i}\}$  is the first pair such that both jobs are tardy in a feasible schedule  $S$ . Consider now applying the Hodgson-Moore algorithm to the job set consisting of all P-jobs except  $\{P_{2i-1}, P_{2i}\}$ , and the R-job. It is easy to see that all jobs  $P_{2j}, j < i$ , are picked as tardy jobs. If we now schedule  $\{P_{2i+2}, P_{2i+1}\}$ , then  $P_{2i+2}$  will still be on time. However, the completion time of  $P_{2i+1}$  is

$$\sum_{j=1}^{i-1} a_{2j-1} + a_{2i+1} + a_{2i+2}.$$

By plugging in (2) and (1), one can easily show that  $P_{2i+1}$  will miss its due date. Since  $P_{2i+2}$  has the largest processing time among all jobs in the current schedule, it will be chosen as a tardy job by the Hodgson-Moore algorithm. Using the same argument, we can show that all even P-jobs are tardy. By assumption,  $P_{2i-1}$  is also a tardy job. Hence, the total number of tardy jobs will be  $n + 1$ , contradicting our assumption that  $S$  is a feasible schedule.

We prove (d) also by contradiction. As we have shown, all the on-time P-jobs must be scheduled before the R-job. The total processing time of these jobs is at least  $\sum_{j=1}^n a_{2j-1}$ . Suppose  $P_m$  is a tardy job scheduled before the R-job, where  $1 \leq m \leq 2n$ . Then the completion time of the R-job would be

$$\begin{aligned} & \sum_{j=1}^n a_{2j-1} + L + a_m \\ &= \left( A - \frac{1}{2} \sum_{j=1}^n (a_{2j} - a_{2j-1}) \right) + L + a_m \end{aligned}$$

$$\begin{aligned}
&= A + L + \left( a_m - \frac{1}{2} \sum_{j=1}^n (a_{2j} - a_{2j-1}) \right) \\
&> A + L .
\end{aligned}$$

Thus, the R-job will miss its due date. By Lemma 3.1,  $S$  can not be a feasible schedule.  $\square$

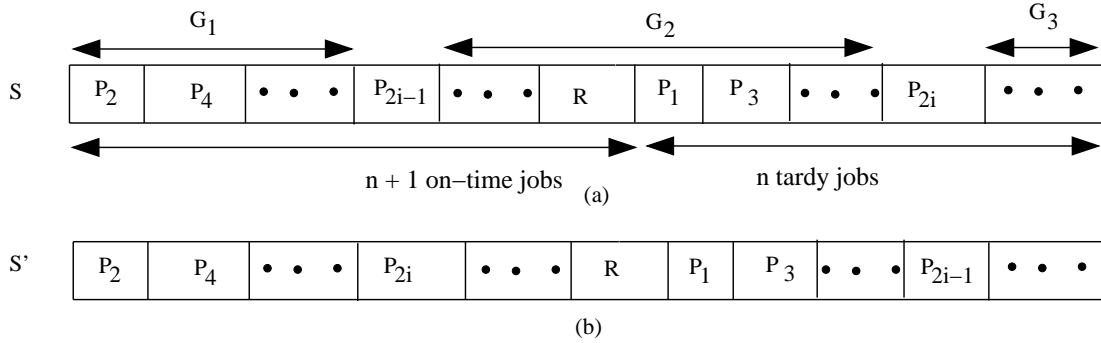


Figure 4: (a) A feasible schedule of the jobs in instance II, (b) The schedule obtained from (a) by interchanging  $P_{2i}$  with  $P_{2i-1}$ .

By Lemma 3.1, we know that in any feasible schedule, all the on-time P-jobs are scheduled before the R-job and all the tardy P-jobs are scheduled after the R-job. It is easy to see that all the on-time P-jobs must be scheduled in increasing order of their due dates; otherwise, some of them will miss their due dates. For the tardy jobs, we can show (by interchange argument) that the total tardiness is minimized by scheduling them in increasing order of their due dates, which is also the same order as their processing times. By Lemma 3.1, there are only two possible choices for each pair  $\{P_{2i-1}, P_{2i}\}$ , either  $P_{2i-1}$  is on time and  $P_{2i}$  is tardy, or  $P_{2i-1}$  is tardy and  $P_{2i}$  is on time.

We now show that in order to minimize the total tardiness, it is always better to pick  $P_{2i-1}$  as a tardy job. Suppose a feasible schedule picks  $P_{2i}$  as a tardy job. We shall prove that by interchanging  $P_{2i}$  with  $P_{2i-1}$ , the total tardiness will be decreased by  $a_{2i} - a_{2i-1}$ . We denote the original schedule as  $S$ , and the new schedule as  $S'$ . We use  $G_1$  to denote the jobs scheduled before  $P_{2i-1}$  in  $S$ ,  $G_2$  to denote the jobs scheduled between  $P_{2i-1}$  and  $P_{2i}$ , and  $G_3$  to denote the jobs scheduled after  $P_{2i}$  in  $S$ ; see Figure 4. Since all jobs scheduled before the R-job are on time, there are exactly  $(i - 1)$  tardy jobs in  $G_2$ .

Note that this interchange will not change the completion time and consequently the tardiness of the jobs in  $G_1$  and  $G_3$ . For each tardy job in  $G_2$ , the tardiness will be increased by  $a_{2i} - a_{2i-1}$ . So, the total increase in tardiness of jobs in  $G_2$  is  $(i - 1)(a_{2i} - a_{2i-1})$ . The completion time of  $P_{2i-1}$  in  $S'$  is the same as the completion time of  $P_{2i}$  in  $S$ . However, since  $d_{P_{2i-1}} = d_{P_{2i}} + i \cdot (a_{2i} - a_{2i-1})$ , the tardiness of  $P_{2i-1}$  in  $S'$  is  $i \cdot (a_{2i} - a_{2i-1})$  less than that of  $P_{2i}$  in  $S$ . In total, the total tardiness is decreased by  $(a_{2i} - a_{2i-1})$ .

On the other hand, in order to ensure that the R-job is on time, we can not pick all the even P-jobs as on-time jobs. Suppose we pick all the even P-jobs. Then the completion time of the R-job will be

$$\sum_{j=1}^n a_{2j} + L = d_R + \frac{1}{2} \sum_{j=1}^n (a_{2j} - a_{2j-1}) ,$$

and hence the R-job will miss its due date. So, we must choose some odd P-jobs as tardy jobs. Let  $B'$  be the total tardiness when we pick all odd P-jobs as tardy jobs. Then,

$$\begin{aligned} B' &= \sum_{i=1}^n T_{P_{2i-1}} \\ &= \sum_{i=1}^n \left( L + \sum_{k=1}^n a_{2k} + \sum_{k=1}^i a_{2k-1} - \left( \sum_{k=1}^i a_{2k} + i \cdot (a_{2i} - a_{2i-1}) \right) \right) \\ &= \sum_{i=1}^n \left( L + \sum_{k=i+1}^n a_{2k} + \sum_{k=1}^i a_{2k-1} - i \cdot (a_{2i} - a_{2i-1}) \right) . \end{aligned}$$

Thus, we have  $B = B' + \frac{1}{2} \sum_{j=1}^n (a_{2j} - a_{2j-1})$ . As we have shown above, each time we interchange  $P_{2i}$  with  $P_{2i-1}$ , the total tardiness will increase by  $a_{2i} - a_{2i-1}$ . At the same time, the completion time of the R-job will decrease by exactly  $a_{2i} - a_{2i-1}$ . So, if we can choose the on-time jobs such that the R-job completes at exactly its due date, then the total tardiness has the minimum value  $B$  among all feasible schedules, and the total processing time of all the on-time P-jobs is exactly  $A$ . This means that there is a solution to the instance of the Even-Odd Partition problem if and only if there is a solution to the instance of the scheduling problem.

From the above discussions, we have the following theorem.

**Theorem 2** *The problem  $1 \parallel \sum T_j \mid \sum U_j$  is binary NP-hard.*

## 4 Conclusion

In this article we have shown that  $1 \parallel C_j \mid \sum U_j$  and  $1 \parallel T_j \mid \sum U_j$  are both binary NP-hard. It is not known whether they are unary NP-hard, or that they can be solved in pseudo-polynomial time. The complexity of  $1 \parallel \max\{T_j\} \mid \sum U_j$  and  $1 \parallel \sum U_j \mid \max\{T_j\}$  remain open. These issues represent major challenges for future research.

**Acknowledgments** The work of the second author is supported in part by the National Science Foundation Grant DMI-0300156. The authors would like to thank George Vairaktarakis for bringing the problems to their attention.

## References

- [1] P. Brucker (2001). *Scheduling Algorithms*, Springer-Verlag, New York.
- [2] C.L. Chen and R.L. Bulfin (1993). Complexity of single machine multi-criteria scheduling problems. *European Journal of Operational Research*, **70**, 115-125.
- [3] C.L. Chen and R.L. Bulfin (1994). Scheduling a single machine to minimize two criteria: maximum tardiness and number of tardy jobs. *IIE Transactions*, **26**, 76-84.
- [4] P. Dileepan and T. Sen (1988). Bicriterion static scheduling research for a single machine. *OMEGA*, **16**, 53-59.
- [5] J. Du and J.Y-T. Leung (1990). Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, **15**, 483-495.
- [6] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, **5**, 287-326.
- [7] M.R. Garey and D.S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.
- [8] M.R. Garey, R.E. Tarjan, and G.T. Wilfong (1988). One-processor scheduling with symmetric earliness and tardiness. *Mathematics of Operations Research*, **13**, 330-348.

- [9] H. Emmons (1975a). One machine sequencing to minimize mean flow time with minimum number tardy. *Naval Research Logistics Quarterly*, **22**, 585-592.
- [10] H. Emmons (1975b). A note on a scheduling problem with dual criteria. *Naval Research Logistics Quarterly*, **22**, 615-616.
- [11] H. Heck and S. Roberts (1972). A note on the extension of a result on scheduling with a secondary criteria. *Naval Research Logistics Quarterly*, **19**, 403-405.
- [12] J.M. Moore (1968). An  $n$  job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, **15**, 102-109.
- [13] S.S. Panwalkar, R.K. Dudek, and M.L. Smith. Sequencing research and the industrial scheduling problem. In *Symposium on the Theory of Scheduling and Its Application*, S.E. Elmaghraby (ed.), Springer-Verlag, New York, 29-38.
- [14] M.L. Pinedo (2002). *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, New Jersey.
- [15] J.G. Shanthikumar (1983). Scheduling  $n$  jobs on one machine to minimize the maximum tardiness with minimum number tardy. *Computers Operations Research*, **10**, 255-266.
- [16] W.E. Smith (1956). Various optimizers for single stage production. *Naval Research Logistics Quarterly*, **3**, 59-66.
- [17] G.L. Vairaktarakis and C-Y. Lee (1995). The single-machine scheduling problem to minimize total tardiness subject to minimum number of tardy jobs. *IIE Transactions*, **27**, 250-256.
- [18] R.E.D. Woolsey (1992). Survival scheduling with Hodgson's rule or see how those salesmen love one another. *INTERFACES*, **22**, 81-84.