Complexity of Scheduling Problems With Generalized Due Dates

C.S. Wong* and Monique Yan
Department of Computer Science
San Francisco State University
San Francisco, CA 94132

Gilbert H. Young
Department of Computer Science
Chinese University of Hong Kong
Shatin, New Territories, Hong Kong

ABSTRACT. We consider the problem of scheduling n independent tasks on a single processor with generalized due dates. The due dates are given according to positions at which jobs are completed, rather than specified by the jobs. We show that the following problems are NP-Complete, $1|\text{prec}, p_j = 1|\sum w_j U_j$, $1|\text{chain}, p_j = 1|\sum w_j U_j$, $1|\text{prec}, p_j = 1|\sum w_j T_j$ and $1|\text{chain}, p_j = 1|\sum w_j T_j$. With the removal of precedence constraints,we prove that the two problems, $1|p_j = 1|\sum w_j U_j$ and $1|p_j = 1|\sum w_j T_j$, are polynomial solvable.

1 Introduction

The Generalized Due Dates (gdd) scheduling model, proposed by Hall [3], is a new class of scheduling problems. In this model, due dates are specified in terms of the position in which a job is completed, rather than specified by the job itself. The generalized due dates did not come out of imagination. Hall [3] and Stecke & Solberg [7] described a number of applications in which the generalized due dates definition is appropriate. These applications include public utility planning problems, survey design, and scheduling problems in some flexible manufacturing system (FMS) en-

*Tel: (415) 338-2858, Fax: (415) 338-6136 and Email: jwong@cs.sfsu.eu

JCMCC 22 (1996), pp. 51-63

vironments. The fundamental point of all problems is that the definition of due dates is independent of particular jobs.

In general, we study the following scheduling problems in this paper: Given a set of n jobs and a set of n generalized due dates, we are to schedule jobs on single processor so as to minimize the total weighted tardiness or total weighted number of late jobs subject to precedence constraints, for example, tree precedence constraint and chain precedence constraint. Under the assumption of generalized due dates (each due date is specific according to the position in which a job is completed), let $d_{[1]}, d_{[2]}, \ldots, d_{[n]}$ be the generalized due dates, where $d_{[1]} \leq d_{[2]} \leq \cdots \leq d_{[n]}$. The scheduling objective is to complete at least i jobs by $d_{[i]}$, $1 \leq i \leq n$. As to which job to be finished at a specific time is irrelevant under the generalized due dates definition.

The outline of the paper is as follows. In the next section, we introduce the notation and show an example of generalized due dates scheduling problems. Section 3 deals with the total weighted number of late jobs problem. The complexity of this problem subject to the arbitrary precedence was asked in [4]. We answer this question by showing that the problem subject to the precedence constraint is NP-Complete, even when the precedence relation is chain. In Section 4, we investigate the total weighted tardiness problem. The complexity of this problem subject to the arbitrary precedence was also asked in [4]. We show that the problem under the precedence constraint is NP-Complete, even when the precedence relation is chain. Finally, we draw some conclusions in the last section.

2 Notation

The terminology used in this paper follows the classification $\alpha|\beta|\gamma$ of scheduling problems suggested by Graham et al. (1979) [2], where α indicates the machine environment (e.g. $\alpha = 1$), β denotes the job characteristics or restrictive requirements (e.g., chain precedence constraint), and, γ is the optimality criterion (e.g., finish time, mean flowtime. maximum lateness, etc.). We also make use of the following notation:

p_j = Processing time of job j

w_j = Weight (value) of job j

$w_{[j]}$ = Weight (value) of the jth job to be completed

r_j = Ready time of job j

C_j = Completion time of job j

$C_{[j]}$ = Completion time of the jth job to be completed

$d_{[j]}$ = Due date of the jth job to be completed for the gdd version

$U_{[j]} = \max\{0, C_{[j]} - d_{[j]}\}$, the tardiness of the jth job to be completed

$$U_{[j]} = \begin{cases} 1 & \text{if } C_{[j]} > d_{[j]} \\ 0 & \text{otherwise} \end{cases}, \text{ the unit penalty of the } j\text{th job to be completed}$$

We define the optimality criterion as the minimization of: (a) the total weighted number of late jobs, $\sum_{j=1}^{n} w_{[j]}U_{[j]}$ and (b) the total weighted tardiness $\sum_{j=1}^{n} w_{[j]}T_{[j]}$. For simplifying the writing of formulas, we will use $\sum w_jU_j$ and $\sum w_jT_j$ for $\sum_{j=1}^{n} w_{[j]}U_{[j]}$ and $\sum_{j=1}^{n} w_{[j]}T_{[j]}$ from here onwards.

There are no assumptions made in this paper about the due dates $d_{[j]}$, $1 \le j \le n$, of generalized due dates, except the very natural assumption $d_{[1]} \le d_{[2]} \le \cdots \le d_{[n]}$. That is, the due date of the ith job to be completed is no greater than that of the jth job to be completed, for $1 \le i < j \le n$.

A precedence constraint is given in the problem. A precedence relation $J_i \rightarrow J_k$ between two jobs means job J_i is required to be completed before job J_k can start. A *feasible schedule* is a sequence that satisfies the precedence constraint. An *optimal schedule* is a feasible schedule such that the value of optimality criteria is minimized.

2.1 Example

In order to illustrate the use of notation, let us consider the following simple example. Given five jobs (J_i, r_i, w_i, p_i), gdd set $\{d_{[i]}\}$ and a chain precedence constraint as shown in Figure 1, we give two feasible schedules S and S^* in Figure 2(a) and 2(b). The total weighted late jobs $(\sum w_jU_j)$ are 100 and 30 for S and S^*, respectively. In this example, S^* is an optimal schedule.

i	1	2	3	4	5
(J_i, r_i, w_i, p_i)	$(J_1, 0, 40, 2)$	$(J_2, 2, 10, 2)$	$(J_3, 2, 30, 3)$	$(J_4, 0, 50, 1)$	$(J_5, 1, 20, 2)$
$d_{[i]}$	1	3	3	8	9
chain precedence constraint: $J_1 \rightarrow J_2 \rightarrow J_3$ and $J_4 \rightarrow J_5$					

Figure 1. Example illustrating the use of notation

$d_{[j]}$	0	1	3	3	8	9
S:		$(J_1, 0, 40, 2)$	$(J_2, 2, 10, 2)$	$(J_3, 2, 30, 3)$	$(J_4, 0, 50, 1)$	$(J_5, 1, 20, 2)$
Time	0	2	4	7	8	10
$U_{[j]}$		1	1	1	0	1
$w_{[j]}U_{[j]}$		40	10	30	0	20

Figure 2(a). Feasible schedule S with $\sum w_jU_j = 100$

$d_{[j]}$	0	1	3	3	8	9
S^*:	$J_4, 0, 50, 1)$	$J_1, 0, 40, 2)$	$J_2, 2, 10, 2)$	$J_3, 2, 30, 3)$	$J_5, 1, 20, 2)$	
Time	0	1	3	5	8	10
$U_{[j]}$	0	0	1	0	1	
$w_{[j]}U_{[j]}$	0	0	10	0	20	

Figure 2(b). Optimal schedule S^* with $\sum w_j U_j = 30$

3 Weighted Number of Late Jobs

The problem of minimizing the total weighted number of late jobs under the generalized due date definition differs from the classical version in that the due dates are specified according to the number of jobs to be completed at a certain moment in time, instead of being specified by particular jobs. The generalized due date definition is less restrictive than the classical due date definition, and the resulting problems can be easier or more difficult to solve than their classical counterparts, and in some cases, the complexity of the problem under the generalized due date definition remains the same. For example, $1|\text{prec}, p_j = 1|\sum U_j$ [3], [5] is generally NP-Hard under the classical due date definition, but becomes polynomial solvable under the generalized due date definition.

Given a set of n tasks, a set of n generalized due dates and a precedence constraint, the problem of scheduling n jobs on a single machine to minimize the total weighted number of late jobs (WU) can be stated formally as follows. Let WU_s denote the total weighted number of late jobs of schedule S. Our goal is to find a feasible non-preemptive schedule S such that WU_s is minimized. Such a schedule will be called an optimal schedule.

In the next section, we will prove that $1|\text{chain}, p_j = 1|\sum w_j U_j$ and $1|\text{prec}, p_j = 1|\sum w_j U_j$ are NP-Complete. Then, in section 3.2, we will show that by removing precedence constraints, the problem $1|p_j = 1|\sum w_j U_j$ is solvable in $O(n \log n)$.

3.1 Jobs with Precedence Constraints

In this section, we will show that the problem $1|\text{chain}, p_j = 1|\sum w_j U_j$ is NP-Complete. The proof is given by transforming the problem from a known NP-Complete problem, 3-Partition [1].

3-Partition. *Given positive integers n, B and a set of integers $A = \{a_1, a_2, \ldots, a_{3n}\}$ with $\sum_{i=1}^{3n} a_i = nB$ and $\frac{B}{4} < a_i < \frac{B}{2}$ for $1 \leq i \leq 3n$, does there exist a partition of A into three element sets $\{A_1, A_2, \ldots, A_n\}$ such that $\sum_{a_i \in A_j} a_i = B$, $1 \leq j \leq n$?*

Theorem 1. *$1|\text{chain}, p_j = 1|\sum w_j U_j$ is NP-Complete.*

Proof: We restate the optimization problem $1|\text{chain}, p_j = 1|\sum w_j U_j$ as the following decision problem (**PWU**): given a set of jobs $J = \{J_i \mid 1 \leq i \leq N\}$ to be processed on a single machine, each job J_i having unit processing time, weight w_i, and the set of generalized due dates $D = \{d_{[1]}, d_{[2]}, \ldots, d_{[N]}\}$, is there a schedule S for J whose total weighted late jobs WU_s is not more than a given value WU_0, i.e. $WU_S \leq WU_0$?

We begin by describing a reduction from the **3-Partition** problem to the problem **PWU**. Let n, B and $A = \{a_1, a_2, \ldots, a_{3n}\}$ be an arbitrary instance of the **3-Partition** problem define above, construct an instance of the **PWU** problem as follows:

a set of $N = 2nB$ jobs:

$$J = \{j_{i,j} \mid 1 \leq j \leq 2a_i, 1 \leq i \leq 3n\}$$

the processing time of jobs:

$$p_{i,j} = 1, 1 \leq j \leq 2a_i \text{ and } 1 \leq i \leq 3n$$

the weight of jobs:

$$w_{i,j} = \begin{cases} 1 & 1 \leq j \leq a_i \\ 0 & a_i + 1 \leq j \leq 2a_i \end{cases}$$

the precedence constraints ($3n$ chains):

$$J_{i,1} \rightarrow J_{i,2} \rightarrow \cdots \rightarrow J_{i,a_i} \rightarrow J_{i,a_i+1} \rightarrow \cdots \rightarrow J_{i,2a_i}, 1 \leq i \leq 3n$$

generalized due dates set:

$$D = \{d_{[j]} \mid d_{[j]} = 2KB + B \text{ for } K = \left\lceil \frac{j}{2B} \right\rceil - 1 \text{ and } 1 \leq j \leq 2nB\},$$

where $\lceil x \rceil$ is the smallest integer greater than or equal to x.

In other words,

$$D = \{d_{1,1}, d_{1,2}, \ldots, d_{1,2B}, \ldots, d_{i,1}, d_{i,2}, \ldots, d_{i,2B}, \ldots, d_{n,1}, d_{n,2}, \ldots, d_{n,2B}\}$$
where $d_{i,j} = (2i-1)B, 1 \leq i \leq n, 1 \leq j \leq 2B,$

and $WU_0 = 0$.

That is, for each integer $a_i \in A$ $(1 \leq i \leq 3n)$, there is a chain $J_{i,1} \rightarrow J_{i,2} \rightarrow \cdots \rightarrow J_{i,a_i} \rightarrow J_{i,a_i+1} \rightarrow \cdots \rightarrow J_{i,2a_i}$ of jobs with unit processing time, weight $w_{i,j} = 1$ for $1 \leq j \leq a$ and $w_{i,j} = 0$ for $a_i + 1 \leq j \leq 2a_i$.

It is easy to see that the decision problem **PWU** is in NP, since a nondeterministic Turing machine can guess an order of all jobs in S and verify in polynomial time that WU_s is not more than WU_0.

We claim that **3-Partition** problem has solution if and only if there exists a feasible schedule S for the job set J with value $WU_s \leq WU_0$, where $WU_0 = 0$.

Suppose that a partition exists, that is, we can re-label the integers $a_1, a_2, \ldots, a_{3n}$ using a_k^i, $1 \leq k \leq 3$, $1 \leq i \leq n$, such that $\sum_{k=1}^{3} a_k^i = B$, $1 \leq i \leq n$. We also re-label the job set $J = \{J_{k,j}^i \mid 1 \leq j \leq 2a_k^i, 1 \leq k \leq 3, 1 \leq i \leq n\}$. Let us consider the following sequence of all tasks of J in schedule S:

$$S = [J_{1,1}^i, J_{1,2}^i, \ldots, J_{1,a_1^i}^i, J_{2,1}^i, J_{2,2}^i, \ldots, J_{2,a_1^i}^i, J_{3,1}^i, J_{3,2}^i, \ldots, J_{3,a_3^i}^i, J_{1,a_1^i+1}^i$$
$$, \ldots, J_{1,2a_1^i}^i, J_{2,a_2^i+1}^i, \ldots, J_{2,2a_1^i}^i, J_{3,a_3^i+1}^i, \ldots, J_{3,2a_3^i}^i \mid i = 1, 2, \ldots, n]$$

First, a feasible schedule S with value $WU_S = 0$ is obtained as shown in Figure 3. Note that, in every time interval $[2B(i-1), 2Bi]$ $(1 \leq i \leq n)$ of schedule S, the first B unit weight jobs $\{J_{1,1}^i, J_{1,2}^i, \ldots, J_{1,a_1^i}^i, J_{2,1}^i, J_{2,2}^i, \ldots, J_{2,a_2^i}^i, J_{3,1}^i, J_{3,2}^i, \ldots, J_{3,a_3^i}^i\}$ are completed on or before the due date $d_{i,j} = (2i-1)B$ $(1 \leq j \leq 2B)$. On the other hand, the remaining B jobs with weight zero $\{J_{1,a_1^i+1}^i, J_{1,a_1^i+2}^i, \ldots, J_{1,2a_1^i}^i, J_{2,a_2^i+1}^i, J_{2,a_2^i+2}^i, \ldots, J_{2,2a_2^i}^i, J_{3,a_3^i+1}^i, J_{3,a_3^i+2}^i, \ldots, J_{3,3a_3^i}^i\}$ are late, they are scheduled in the time interval $[2Bi - B, 2Bi]$ for any i, $1 \leq i \leq n$. Therefore, the weighted number of late jobs of the schedule S, $WU_S = WU_0 = 0$.

Conversely, suppose there exists a feasible schedule S_0 in which $WU_{S_0} = 0$, in other words, all jobs with unit weight in S_0 must be scheduled on-time. Due to our choice of due dates, it is easily seen that all on-time jobs must be scheduled in time intervals $[2B(i-1), 2Bi - B]$ for $1 \leq i \leq n$. This, in turn, implies that $(n-1)B$ late jobs (zero weight jobs) must be scheduled in time intervals $[2Bi - B, 2Bi]$, for $1 \leq i \leq n-1$ and the remaining B late jobs must be scheduled at or after time $2nB - B$. Without affecting the value of WU_{S_0}, we assume that the last B late jobs are scheduled in time interval $[2nB - B, 2nB]$. Hence the jobs have to be processed continuously without any idle time on machine in order to have $WU_{S_0} = 0$. For the simplicity, we call n chains of unit weight jobs $J_{i,1} \rightarrow J_{i,2} \rightarrow \cdots \rightarrow J_{i,a_i}$ as a Ja_i-type jobs for $1 \leq i \leq n$. Each Ja_i job has total of a_i processing time, for $1 \leq i \leq n$. The key idea of the proof here is that each time interval $[2B(i-1), 2Bi]$, for $1 \leq i \leq n$, must contain exactly 3 Ja_i-type jobs and the chains that follow them. It can be seen later that it is sufficient to prove this for the first time interval $[0, 2B]$. Suppose only two Ja_i-type jobs (say Ja_1, Ja_2) finish by the time $2B$. Then the number of unit processing time jobs with weight 1 which finish on or before $2B$ is equal to $(a_1 + a_2)$, which

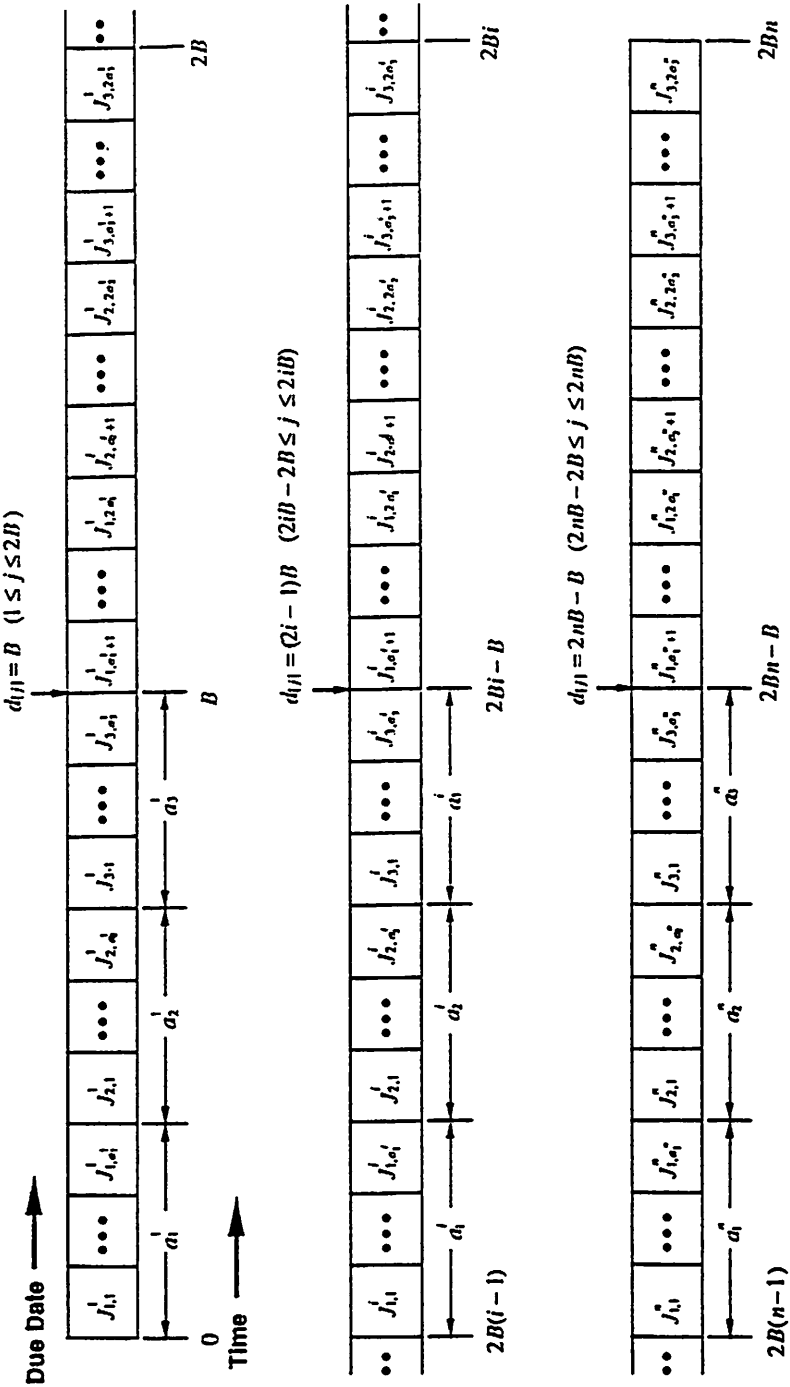


Figure 3. A feasible schedule S with value $WU_S = 0$ for problem PWU

is clearly less than B (since $\frac{B}{4} < a_i < \frac{B}{2}$). Consequently, $WU_{S_0} > 0$. Hence no less than three Ja_i-type jobs finish before $2B$ in S_0. Moreover if four Ja_i-type jobs (say Ja_1, Ja_2, Ja_3, Ja_4) complete on or before the time $2B$, then the number of unit processing time jobs with weight 1 is equal to $(a_1 + a_2 + a_3 + a_4)$ which is more than B. Now it is clear that exactly three jobs of Ja_i-type (say Ja_1, Ja_2, Ja_3) finish on or before the time $2B$. If $a_1 + a_2 + a_3 \neq B$, this implies that the total number of jobs with weight 1 finished on or before B is less than B. Thus $WU_{S_0} > 0$. Therefore, the three Ja_i-type jobs scheduled in the time interval $[0, 2B]$ have the property that $a_1 + a_2 + a_3 = B$. Note that only three Ja_i-type jobs and the chains that follow them are scheduled in this interval (and the total processing time of these jobs is equal to $2B$). Now it is evident that the above proof holds for each time interval $[2B(i-1)2Bi]$, for $1 \leq i \leq n$. Hence there exists a **3-Partition**. This completes the proof of Theorem 1. □

The complexity of the problem $1|\text{prec}, p_j = 1|\sum w_j U_j$ was asked in [4]. We provide the answer as follows.

Corollary 1. *$1|\text{prec}, p_j = 1|\sum w_j U_j$ is NP-Complete.*

Proof: The problem $1|\text{chain}, p_j = 1|\sum w_j U_j$, which was proved in Theorem 1 to be NP-Complete, is a special case of the problem $1|\text{prec}, p_j = 1|\sum w_j U_j$. Therefore, $1|\text{prec}, p_j = 1|\sum w_j U_j$ is also NP-Complete. □

3.2 Jobs without Precedence Constraints

We have shown that the generalized due dates scheduling problem involving (chain) precedence constraints between unit processing time jobs so as to minimize weighted number of late jobs is NP-complete. We will show that removal of precedence constraints from this NP-complete problem results in a problem that is solvable within polynomial time $O(n \log n)$.

For simplicity, assume the first job begin at time $t = 0$, then any given sequence induces a well defined completion time $C_{[j]} = j$, since the processing time of each job is one unit. Also, jobs are executed without interruption and without idle time between them. The idea of constructing the optimal schedule S is as follows:

Algorithm X:
Input: A set of tasks and a set of generalized due dates
Output: An optimal schedule
Method:
(1) $i \leftarrow 1$
(2) $\Gamma \leftarrow \{J_i \mid J_i$ has not been scheduled$\}$
(3) If $\Gamma = \Phi$, then stop.
(4) $T' \leftarrow J_S$, such that $w_S = \min\{w_j \mid J_j \in \Gamma\}$;

$T'' \leftarrow J_L$, such that $w_L = \max\{w_j \mid J_j \in \Gamma\}$.

(5) If $i > d_{[i]}$, then /* if this is a late job */
Schedule T'
else
Schedule T''

(6) $i \leftarrow i + 1$. Return to Step (2).

Example: Consider the following five tasks (J_i, r_i, w_i, p_i): $(J_1, 0, 2, 1)$, $(J_2, 0, 5, 1)$, $(J_3, 0, 1, 1)$, $(J_4, 0, 3, 1)$ and $(J_5, 0, 4, 1)$ and gdd set $\{d_{[i]}\} = \{1, 1, 2, 4, 4\}$. The optimal schedule S obtained from Algorithm X is shown in Figure 4 with $\sum w_j U_j = 6$.

i	1	2	3	4	5
$d_{[i]}$	1	1	2	4	4
$C_{[i]}$	1	2	3	4	5
$U_{[i]}$	0	1	1	0	1
$w_{[i]}U_{[i]}$	0	1	2	0	3

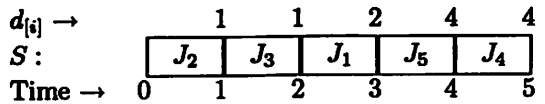


Figure 4. Example illustrating Algorithm X

Theorem 2. *Algorithm X always give an optimal schedule S for the problem $1|p_j = 1|\sum w_j U_j$.*

Proof: We will prove the theorem by contradiction. Assume S is the schedule produced by Algorithm X which is not an optimal schedule, then there must exist a schedule S' such that $WU_{S'} < WU_S$. First, it is easy to see that:

(1) For any arbitrary schedule S^*, the completion time of the jth job in schedule S^*, $C_{[j]}(S^*)$, is always equal to completion time of the jth job in schedule S, $C_{[j]}(S)$, for $1 \leq j \leq n$.

(2) For any arbitrary schedule S^*, the number of late jobs under S^* must be exactly equal to the number of late jobs under S.

Without loss of generality we can assume that jobs are numbered in such a way that J_i and J_j $(i < j)$ implies $w_i \leq w_j$, i.e., we assume that the jobs are indexed according to their weights $w_1 \leq w_2 \leq \cdots \leq w_n$. We also assume that there are x late jobs in both schedule S and S', then the weighted number of late jobs in schedule S will simply be

$$WU_S = \sum_{j=1}^{n} w_j U_j = \sum_{j=1}^{x} w_j 1 + \sum_{j=x+1}^{n} w_j 0 = \sum_{j=1}^{x} w_j$$

On the other hand, the objective value of schedules S' is

$$WU_{S'} = \sum_{j=1}^{n} w'_j U_j = \sum_{j=1}^{x} w'_j 1 + \sum_{j=x+1}^{n} w'_j 0 = \sum_{j=1}^{x} w'_j$$

Consequently, $WU_S \leq WU_{S'}$ since $\sum_{j=1}^{x} w_j$ is the sum of the x smallest weights in job set. It is obviously contradicting our assumption that $WU_{S'} < WU_S$. This completes the proof of theorem 2. □

4 Total Weighted Tardiness

In this section, we shall explore the complexity status of the total weighted tardiness open problems under the generalized due dates definition proposed by Hall [4]. Given a set of n tasks, a set of n generalized due dates and a precedence constraint, the problem of scheduling all n jobs on a single machine to minimize the total weighted tardiness (WT) can be stated formally as follows. Let WT_S denote the total weighted tardiness in a feasible schedule S. Our goal is to find a feasible non-preemptive schedule S such that WT_S is minimized. Such a schedule will be called an optimal schedule.

In the generalized due dates definition, several single machine total weighted tardiness scheduling problems under this new definition of due dates have been investigated. The problem $1|r_j|\sum T_j$ was shown to be NP-Complete while $1||\sum T_j$ was shown to be polynomial solvable by Hall [3]. NP-Completeness for the $1||\sum w_j T_j$ has been established by Sriskandarajah [6]; the case in which there are arbitrary precedence constraints and equal weights, $1|\text{prec}|\sum T_j$, is still NP-Complete. But the case of unit processing time, $1|\text{prec}, p_j = 1|\sum T_j$, is polynomial solvable (Hall et al. [4]) even though the classical version of this problem is NP-Complete.

In the next section, we will prove that $1|\text{prec}, p_j = 1|\sum w_j T_j$ and $1|\text{chain}, p_j = 1|\sum w_j T_j$ are NP-Complete. However, if the precedence constraints are removed, the problem $1|p_j = 1|\sum w_j T_j$ can be solved by means of an $O(n \log n)$ algorithm given in section 4.2.

4.1 Jobs with Precedence Constraints

The complexity of the problem $1|\text{prec}, p_j = 1|\sum w_j T_j$ was asked in [4]. We provide the answer as follows.

Corollary 2. *$1|\text{prec}, p_j = 1|\sum w_j T_j$ and $1|\text{chain}, p_j = 1|\sum w_j T_j$ are NP-Complete.*

Proof: By using the same reduction as in Theorem 1 with $WT_0 = 0$. we can easily show that $1|\text{chain}, p_j = \sum w_j T_j$ is NP-Complete. By using Corollary 1, we can conclude that $1|\text{prec}, p_j = 1|\sum w_j T_j$ is NP-Complete. □

4.2 Jobs without Precedence Constraints

We next consider the total weighted tardiness problem without the precedence constraint. The specific due date version of this problem is easily seen to be a linear assignment problem and thereby solvable in $O(n^3)$ steps, as pointed out by Lenstra and Rinnooy Kan [5]. The generalized due date version of this problem can be solved by ordering the jobs according to the schedule given by Algorithm Y below. The basic idea is to schedule the job with ith largest weight into the time slot with the ith smallest tardiness (for this problem, the tardiness of each job can be calculated by using the given due date only).

Algorithm Y

Input: A set of n jobs and a set of n generalized due dates

Output: An optimal schedule S

Method:

(1) Compute the tardiness of all jobs, $T_{[j]} = \max\{0, j - d_{[j]}\}$ for $1 \leq j \leq n$.

(2) Sort tardiness $T_{[j]}$ in non-decreasing order of their values. We have the sequence $T_{z_1} \leq \cdots \leq T_{z_i} \leq \cdots \leq T_{z_n}$, where z_i is the position of the job with the ith smallest tardiness in S.

(3) Sort jobs a non-increasing order of their weights. We have the sequence $w_{q_1} \geq \cdots \geq w_{q_i} \geq \cdots \geq w_{q_n}$, where q_i is the job with the ith largest weight.

(4) To obtain the final schedule S, schedule job J_{q_i} in position z_i in S for $1 \leq i \leq n$.

Example: We illustrate Algorithm Y with a numerical example. Figure 5 shows all four steps of Algorithm Y for five jobs with weight (J_i, w_i) and generalized due date $d_{[i]}$, $1 \leq i \leq 5$.

Theorem 3. *Algorithm Y always gives an optimal schedule S for the problem* $1|p_j = 1|\sum w_j T_j$.

Proof: The problem can be proved by contradiction. Assume S is the schedule produced by Algorithm Y which is not an optimal schedule, then there must exist an optimal schedule S'. Since S' is not identical S, there must exist at least a pair jobs, J'_{q_i} and $J'_{q_{i+1}}$ in S' such that $w'_{q_i} < w'_{q_{i+1}}$ and $T'_{z_i} \leq T'_{z_{i+1}}$. Now construct a new sequence S^*, in which jobs J'_{q_i} and $J'_{q_{i+1}}$ are interchanged and the sequence for the other $(n-2)$ jobs remain the same as in S'. It is obvious that S' and S^* differ only in the weighted

tardiness of jobs J'_{q_i} and $J'_{q_{i+1}}$. Hence

$$\begin{aligned} WT_{S^*} - WT_{S'} &= \left[T'_{z_i} w'_{q_{i+1}} + T'_{z_{i+1}} w'_{q_i}\right] - \left[T'_{z_i} w'_{q_i} + T'_{z_{i+1}} w'_{q_{i+1}}\right] \\ &= \left(w'_{q_i} - w'_{q_{i+1}}\right)\left(T'_{z_{i+1}} - T'_{z_i}\right) \\ &\leq 0 \end{aligned}$$

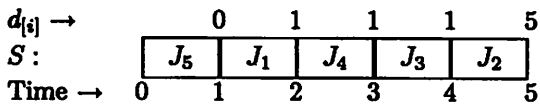
This means that the interchange of jobs J'_{q_i} and $J'_{q_{i+1}}$ does not increase the value of total weighted tardiness of schedule S'. We can repeat the same procedure until we cannot find a pair of two jobs such that the above condition holds. The final schedule is exactly the schedule S. Therefore $WT_S \leq WT_{S'}$, S is also an optimal schedule. □

i	1	2	3	4	5
(J_i, w_i)	$(J_1, 5)$	$(J_2, 8)$	$(J_3, 2)$	$(J_4, 3)$	$(J_5, 6)$
$d_{[i]}$	0	1	1	1	5
$C_{[i]}$	1	2	3	4	5
$T_{[i]}$	1	1	2	3	0

$$T_{z_1} \leq T_{z_2} \leq T_{z_3} \leq T_{z_4} \leq T_{z_5} = T_{[5]} \leq T_{[1]} \leq T_{[2]} \leq T_{[3]} \leq T_{[4]}$$

$$w_{q_1} \geq w_{q_2} \geq w_{q_3} \geq w_{q_4} \geq w_{q_5} = w_2 \geq w_5 \geq w_1 \geq w_4 \geq w_3$$

(a) After step 1, 2 and 3.



(b) After step 4. The optimal schedule is S.

Figure 5. Example illustrating Algorithm Y

5 Conclusion

In this paper we have investigated several single machine scheduling problems under the generalized due dates definition. We now summarize the results and discuss some possible directions for future research for each problem.

In section 3 we have considered the problem of minimizing the weighted number of late jobs. Several versions of this problem were studied. We showed that the problems $1|\text{prec}, p_j = 1|\sum w_j U_j$ and $1|\text{chain}, p_j = 1|\sum w_j U_j$ are NP-Complete. This answers the question that was posted in [4]. When

the precedence constraint is removed, we presented an $O(n \log n)$ algorithm to solve the problem $1|p_j = 1|\sum w_j U_j$.

In section 4, we investigated the problem of minimizing the total weighted tardiness. We showed that the problems $1|\text{prec}, p_j = 1|\sum w_j T_j$ and $1|\text{chain}, p_j = 1|\sum w_j T_j$ are NP-Complete. This answers the question that was asked in [4]. When the precedence constraint is removed, we presented an $O(n \log n)$ algorithm to solve the problem $1|p_j = 1|\sum w_j T_j$.

For future directions, it will be beneficial to consider heuristic algorithms for problems proved to be NP-Complete

References

[1] M.R. Garey and D.S. Johnson , *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman & Company, 1979.

[2] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan, Optimization and approximation in Deterministic Sequencing and Scheduling: A Survey, *Annals of Discrete Mathematics* **5** (1979), 287–326.

[3] N.G. Hall, Scheduling Problems with Generalized Due Dates, *IIE Transactions* **18** (1986), 220–222.

[4] N.G. Hall, S.P. Sethi and C. Sriskandarajah, On the Complexity of generalized Due Date Scheduling Problems. *European Journal of Operational Research* **51, N1** (1991), 100–109.

[5] J,K, Lenstra and A.H.G. Rinnooy Kan, Complexity of Scheduling under Precedence Constraints, *Operations Research* **26** (1978), 22–35.

[6] C. Sriskandarajah, A Note on the Generalized Due Dates Scheduling Problem, *Naval Research Logistics* **37, N4** (1990), 587–597.

[7] K.E. Stecke and J.J. Solbert, Loading and control Policies for a Flexible Manufacturing System, *International of Production Research* **19** (1981), 481–490.