

Short communication

A single-machine scheduling with generalized due dates to minimize total weighted late work

Abstract

In the paper, we consider a single-machine scheduling problem with generalized due dates, in which the objective is to minimize total weighted work. This problem was proven to be NP-hard by Mosheiov et al. [8]. However, the exact complexity remains open. We show that the problem is strongly NP-hard, and is weakly NP-hard if the lengths of the intervals between the consecutive due dates are identical.

Keywords: Scheduling; total late work; generalized due dates; computational complexity

1. Introduction

Consider a scheduling problem such that the due date is assigned not to the specific job but to the job position. Such a due date is referred to as the *generalized due date (GDD)*. Since the scheduling problem with GDD was initiated from Hall [5], much research has been done in [2, 4, 6, 9, 10, 11]. Recently, Mosheiov et al. [8] considered single-machine scheduling problems with GDD to minimize total late work. They showed that the problem can be solved by the Shortest Processing Time first (SPT) rule, while it is NP-hard if each job has a different weight. Note that it is unknown whether the case with

the different weights is strongly NP-hard or not. We establish the exact complexity for the case with the different weights.

The remainder of this paper is organized as follows. Sections 2 and 3 defines the problem formally and establishes the computational complexity.

2. Problem definition

Our problem can be formally stated as follows: For each job $j \in \mathcal{J} = \{1, 2, \dots, n\}$, let p_j and w_j be the processing time and the weight, respectively. Let $\pi = (\pi(1), \pi(2), \dots, \pi(n))$ be a schedule, where $\pi(j)$ is the j th job. For each $j \in \mathcal{J}$, let $S_j(\pi)$ and $C_j(\pi)$ be the start and completion times of job j in π , respectively, and $\pi^{-1}(j)$ be the position of job j in π . In our model, unlike the traditional scheduling problem, the due date d_i is assigned not to the specific job, but to the job positioned i th for each due date $i \in \mathcal{D} = \{1, 2, \dots, n\}$. For simplicity, assume that $d_0 = 0$ and

$$d_1 \leq d_2 \leq \dots \leq d_n.$$

GDD has two special cases depending on the condition of the due dates. The first and the second cases have a common due date with

$$d_i = d \text{ for } i \in \mathcal{D}, \quad (1)$$

and identical lengths of the intervals between the consecutive due dates, that is,

$$d_i = i\delta \text{ and } d_i - d_{i-1} = \delta \text{ for } i \in \mathcal{D}, \quad (2)$$

respectively. Let the due dates with relations (1) and (2) be referred to as the *common due dates* (CDD) and *periodic due dates* (PDD), respectively. For each $j \in \mathcal{J}$, let $T_j(\pi)$ and $Y_j(\pi)$ be the tardiness and late work of a job j in π , respectively, which are calculated as

$$T_j(\pi) = \max\{0, L_j(\pi)\} \text{ and } Y_j(\pi) = \min\{p_j, T_j(\pi)\},$$

where $L_j(\pi) = C_j(\pi) - d_{\pi^{-1}(j)}$. The objective is to find a schedule π to minimize total weighted late work, which is calculated as

$$z(\pi) = \sum_{j \in \mathcal{J}} w_j Y_j(\pi).$$

We follow the standard three-field notation $1|\beta|\sum_{j \in \mathcal{J}} w_j Y_j$ introduced by Graham et al. [3], where $\beta \in \{CDD, PDD, GDD\}$ describes the characteristics of the due dates. This paper establishes the complexities of three cases.

Table 1 summarizes our results (note that ‘wNP-hard’ and ‘sNP-hard’ stand for weakly and strongly NP-hard, respectively).

Table 1: Complexity for $1|\beta|\gamma$

$\gamma \setminus \beta$	<i>CDD</i>	<i>PDD</i>	<i>GDD</i>
$\sum w_j T_j$	wNP-hard [7, 11]	wNP-hard [2]	sNP-hard [4]
$\sum w_j Y_j$	polynomially solvable [8]	wNP-hard (Cor. 1)	sNP-hard (Thm. 1)

3. Computational complexity

In this section, we show that $1|GDD|\sum w_j Y_j$ and $1|PDD|\sum w_j Y_j$ are strongly and weakly NP-hard, respectively.

Theorem 1. $1|GDD|\sum w_j Y_j$ is strongly NP-hard.

Proof Gao and Yuan [4] showed that $1|GDD|\sum w_j T_j$ is strongly NP-hard. It is observed from the reduced instance in their proof that $T_j = Y_j$ holds for each job $j \in \mathcal{J}$ in the optimal schedule. Thus, $1|GDD|\sum w_j Y_j$ is strongly NP-hard. ■

Theorem 2. $1|PDD|\sum w_j Y_j$ is NP-hard.

Proof For simplicity, for $1|CDD|\sum w_j T_j$, let \bar{p}_j and \bar{w}_j be the processing time and weight of job $j \in \{1, 2, \dots, n\}$, respectively, and d be the common due date. Yuan [11] showed that $1|CDD|\sum w_j T_j$ is NP-hard, even if

$$\sum_{j=1}^n \bar{p}_j \leq 2d + 1. \quad (3)$$

Given an instance of $1|CDD|\sum w_j T_j$, we can construct an instance of $1|PDD|\sum w_j Y_j$ with $(n + 1)$ jobs in $\mathcal{J} = \{0, 1, \dots, n\}$ such that

- $p_0 = 0$ and $w_0 = 1 + \sum_{j=1}^n \bar{w}_j$;
- $p_j = d + \bar{p}_j$ and $w_j = \bar{w}_j$, $j = 1, 2, \dots, n$;
- $\delta = d$.

It is observed that job 0 is processed at the first position in any optimal schedule for the reduced instance of $1|PDD|\sum w_j Y_j$. Thus, we consider only a schedule π for the reduced instance with $\pi(1) = 0$, that is, a schedule $\pi = (0, \bar{\pi})$, where $\bar{\pi}$ is the schedule for a given instance of $1|CDD|\sum w_j T_j$. Note that the k th job in $\bar{\pi}$ is the $(k+1)$ th job in π . Then, we have

$$C_{\pi(k+1)}(\pi) = \sum_{h=2}^{k+1} p_{\pi(h)} = \sum_{h=1}^k (d + p_{\bar{\pi}(h)}) = kd + C_{\bar{\pi}(k)}(\bar{\pi}), \quad (4)$$

where the first equality holds due to $p_{\pi(1)} = 0$. If job j is the k th job in $\bar{\pi}$, then we have, by equation (4),

$$L_j(\pi) = kd + C_{\bar{\pi}(k)}(\bar{\pi}) - (k+1)\delta = C_j(\bar{\pi}) - d = L_j(\bar{\pi})$$

and

$$T_j(\pi) = T_j(\bar{\pi}).$$

By inequality (3), we have $T_j(\bar{\pi}) \leq \sum_{j=1}^n \bar{p}_j - d \leq d+1 \leq d + \bar{p}_j$. Then

$$Y_j(\pi) = \min\{p_j, T_j(\pi)\} = \min\{d + \bar{p}_j, T_j(\bar{\pi})\} = T_j(\bar{\pi}).$$

Since job 0 is not tardy in π and $w_j = \bar{w}_j$, $j = 1, 2, \dots, n$, the objective values of the two schedules π and $\bar{\pi}$ in each instance are the same. This implies that $1|CDD|\sum w_j T_j$ is special case of $1|PDD|\sum w_j Y_j$. Thus, Theorem 2 holds. ■

Let a job j be referred to as *small* if $p_j \leq \delta$, and *large*, otherwise. Let \mathcal{S} and \mathcal{L} be the sets of small and large jobs, respectively. Let

$$a_j = \begin{cases} \delta - p_j & \text{for } j \in \mathcal{S} \\ p_j - \delta & \text{for } j \in \mathcal{L}. \end{cases}$$

Furthermore, let a_j be referred to as *auxiliary processing time* for $j \in \mathcal{L}$. Under a schedule π , let a job j be referred to as *early* if $Y_j(\pi) = 0$, *partially late* if $0 < Y_j(\pi) < p_j$, and *fully late* if $Y_j(\pi) = p_j$.

Observation 1. In $1|PDD|\sum w_j Y_j$, an optimal schedule π can be represented as

$$\pi = (\pi_s, \pi_e, \pi_p, \pi_f),$$

where π_s , π_e , π_p and π_f are sequences of small, early, partially late, and fully late jobs, respectively. Furthermore, the jobs in π_i for $i \in \{s, e, f\}$ are sequenced arbitrarily.

By Observation 1, henceforth, we construct only a schedule for large jobs. Let $d = \sum_{j \in \mathcal{S}} a_j$ and $[h]$ be the h th large job in π . Note that

$$T_{[h]}(\pi) = \max \left\{ 0, \sum_{i=1}^h a_{[i]} - d \right\} \quad \text{and} \quad Y_{[h]}(\pi) = \min \{ p_{[h]}, T_{[h]}(\pi) \}. \quad (5)$$

Let \mathcal{P} and x be the set of partially late jobs and the first partially late job in the optimal schedule, respectively. Let x be referred to as a *straddling* job.

Lemma 1. *In an optimal schedule π , jobs in $\mathcal{P} \setminus \{x\}$ are sequenced in non-increasing order of w_j/a_j .*

Proof Suppose that there exist two jobs $i = [k]$ and $j = [k+1]$ in $\mathcal{P} \setminus \{x\}$ with

$$\frac{w_i}{a_i} < \frac{w_j}{a_j}. \quad (6)$$

Note that by $[k-1] \in \mathcal{P}$, $T_{[k-1]}(\pi) > 0$. Then, by $\{i, j\} \subset \mathcal{P}$ and (5),

$$w_i Y_i(\pi) + w_j Y_j(\pi) = w_i (T_{[k-1]}(\pi) + a_i) + w_j (T_{[k-1]}(\pi) + a_i + a_j). \quad (7)$$

Let $\bar{\pi}$ be the schedule constructed by interchanging the positions of jobs i and j from π . Then,

$$w_j Y_j(\bar{\pi}) + w_i Y_i(\bar{\pi}) \leq w_j (T_{[k-1]}(\pi) + a_j) + w_i (T_{[k-1]}(\pi) + a_j + a_i). \quad (8)$$

By (6)-(8), we have

$$z(\pi) - z(\bar{\pi}) \geq w_j a_i - w_i a_j > 0.$$

This contradicts to the optimality of π . ■

Theorem 3. $1|PDD|\sum w_j Y_j$ can be solved in pseudo-polynomial time.

Proof We present a DP based on Observation 1 and Lemma 1. Suppose that in an optimal schedule, the auxiliary processing time and the weight of the straddling job x are a and w , respectively. Renumber the remaining large jobs such that

$$\frac{w_1}{a_1} \geq \frac{w_2}{a_2} \geq \dots \geq \frac{w_m}{a_m},$$

where $m = |\mathcal{L}| - 1$. Then, we construct a schedule of jobs in $\{1, 2, \dots, m\}$ by applying Algorithm 1. For each $k \in \{1, 2, \dots, m\}$, the k th phase of Algorithm 1 produces a set \mathcal{S}_k of states. Each state in \mathcal{S}_k is expressed as a vector $S = [s_1, s_2, s_3, s_4, s_5]$ representing the information of a partial schedule for the first k jobs, where

- The component s_1 is total auxiliary processing time of early jobs;
- The components s_2 and s_3 are total auxiliary processing time and total weight of partially late jobs, respectively;
- The component s_4 is the last partially late job in the current partial schedule;
- The component s_5 is total weighted late work of a partial schedule.

The initial set \mathcal{S}_0 contains only one state $[0, 0, 0, 0, 0]$. For each $k \in \{1, 2, \dots, m\}$, \mathcal{S}_k is obtained from \mathcal{S}_{k-1} through three mappings, F_1 , F_2 , and F_3 , which translate $S := [s_1, s_2, s_3, s_4, s_5] \in \mathcal{S}_{k-1}$ into the states in \mathcal{S}_k as follows:

i) Calculate F_1 defined by

$$F_1(a_k, w_k, S) = [s_1, s_2, s_3, s_4, s_5 + w_k(a_k + \delta)].$$

Note that job k becomes a fully late job through mapping F_1 ;

ii) Calculate F_2 defined by

$$F_2(a_k, w_k, S) = [s_1, s_2 + a_k, s_3 + w_k, k, s_5 + w_k(s_2 + a_k)].$$

Note that job k becomes a partially late job through mapping F_2 ;

iii) If $s_1 + a_k < d$, then calculate F_3 defined by

$$F_3(a_k, w_k, S) = [s_1 + a_k, s_2, s_3, s_4, s_5].$$

Note that job k becomes an early job through mapping F_3 .

After completing the m th phase, we place the straddling job x if jobs x and s_4 can be the first and last partially late jobs, respectively. That is, shift all (partially and fully) late jobs to the right by $(s_1 + a - d)$ and insert the straddling job x on interval $[s_1, s_1 + a]$ if the state $S \in \mathcal{S}_m$ belongs to the following set from (5):

$$\mathcal{Q} = \{S \in \mathcal{S}_m \mid s_1 \leq d < s_1 + a \text{ and } \delta \leq s_1 + a + s_2 - d < a_{s_4} + \delta\}.$$

At this time, total weighted late work of a feasible schedule is calculated as

$$G(S) = s_5 + (s_3 + w)(s_1 + a - d) \quad \text{for } S \in \mathcal{Q}.$$

Algorithm 1 outputs a schedule with the minimum $G(S)$ among $S \in \mathcal{Q}$.

Algorithm 1: DP for $1|PDD|\sum w_j Y_j$ with a fixed straddling job

```

1  $\mathcal{S}_0 \leftarrow \{[0, 0, 0, 0, 0]\};$ 
2 for  $k \leftarrow 1$  to  $m$  do
3   for each  $S := [s_1, s_2, s_3, s_4, s_5] \in \mathcal{S}_{k-1}$  do
4      $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup F_1(a_k, w_k, S) \cup F_2(a_k, w_k, S) \cup F_3(a_k, w_k, S);$ 
5   end
6 end
7  $\mathcal{Q} = \{S \in \mathcal{S}_m \mid s_1 \leq d < s_1 + a \text{ and } \delta \leq s_1 + a + s_2 - d < a_{s_4} + \delta\};$ 
8 for each  $S := [s_1, s_2, s_3, s_4, s_5] \in \mathcal{Q}$  do
9    $G(S) \leftarrow s_5 + (s_3 + w)(s_1 + a - d);$ 
10 end
11 return  $\min\{G(S) \mid S \in \mathcal{Q}\};$ 

```

Note that the number of states in the algorithm is bounded by $O(lA^2WT)$, where $l = |\mathcal{L}|$, $A = \sum_{j \in \mathcal{L}} a_j$, $W = \sum_{j \in \mathcal{L}} w_j$, and $T = \sum_{j \in \mathcal{L}} w_j p_j$. Hence, Algorithm 1 is a pseudo-polynomial algorithm. Since the possible number of straddling job is l , $1|PDD|\sum w_j Y_j$ can be solved in pseudo-polynomial time. ■

Corollary 1. $1|PDD|\sum w_j Y_j$ is weakly NP-hard.

Proof It immediately holds by Theorems 2 and 3. ■

References

- [1] R.A. Ahuja, K. Mehlhorn, J.B. Orlin (1993) *Network Flows: Theory, Algorithms, and Applications*. Prince Hall, Upper Saddle River, NJ.
- [2] B.-C. Choi, K.M. Kim, Y. Min, M.-J. Park, Scheduling with generalized and periodic due dates under single- and two-machine environments, Working paper.
- [3] 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*, 3(1979), 287–326.
- [4] Y. Gao and J.J. Yuan, Unary NP-hardness of minimizing total weighted tardiness with generalized due dates, *Operations Research Letters*, 44(2016), 92–95.
- [5] N.G. Hall, Scheduling problems with generalized due dates, *IIE Transactions*, 18(1986), 220–222.
- [6] N.G. Hall, S.P. Sethi and C. Sriskandarajah, On the complexity of generalized due date scheduling problems, *European Journal of Operational Research*, 51(1991), 100–109.
- [7] E.L. Lawler and J.M. Moore, A functional equation and its application to resource allocation and sequencing problems, *Management Science*, 16(1969), 77–84.

- [8] G. Mosheiov, D. Oron, D. Shabtay, Minimizing total late work on a single machine with generalized due dates, *European Journal of Operational Research*, <https://doi.org/10.1016/j.ejor.2020.12.061>.
- [9] M.-J. Park, B.-C. Choi, K.M. Kim, Y. Min, Two-machine ordered flow shop scheduling with generalized due dates, *Asia-Pacific Journal of Operational Research*, 37(2020) 1–16.
- [10] S. Srikandarajah. A note on the generalized due dates scheduling problem, *Naval Research Logistics*, 37(1990), 587–597.
- [11] J.J. Yuan, The NP-hardness of the single machine common due date weighted tardiness problem, *Journal of Systems Science and Complexity* 5(1992), 328–333.