Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 374 (2007) 159–169
Theoretical Computer Science
www.elsevier.com/locate/tcs

Scheduling jobs with agreeable processing times and due dates on a single batch processing machine

L.L. Liu, C.T. Ng*, T.C.E. Cheng

Department of Logistics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

Received 26 October 2006; received in revised form 11 December 2006; accepted 22 December 2006

Communicated by D.-Z. Du

Abstract

In this paper we study the problems of scheduling jobs with agreeable processing times and due dates on a single batch processing machine to minimize total tardiness, and weighted number of tardy jobs. We prove that the problem of minimizing total tardiness is NP-hard even if the machine capacity is two jobs and we develop a pseudo-polynomial-time algorithm for an NP-hard special case of this problem. We also develop a pseudo-polynomial-time algorithm for the NP-hard problem of minimizing weighted number of tardy jobs, which suggests that this problem cannot be strongly NP-hard unless P = NP.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Scheduling; Batch processing; Agreeable

1. Introduction

Batch scheduling has attracted wide attention of the scheduling research community over the past two decades. This scheduling model is primarily motivated by the burn-in operations in the final testing stage of very large-scale integrated circuits manufacture. A batch processing machine can process several jobs simultaneously. The processing time of a batch is equal to the longest processing time of the jobs in the batch. All the jobs contained in the same batch start and complete at the same time. Once processing of a batch is started, it cannot be interrupted, nor can other jobs be added to the batch. Besides application in very large-scale integrated circuits manufacture, batch scheduling can also be found in a variety of other manufacturing environments such as heat treatment in the metalworking industry, and diffusion or oxidation in wafer fabrication of semiconductor manufacture.

Du and Leung [4] proved that the traditional scheduling problem of minimizing total tardiness on a single machine is NP-hard. Lawler [8] developed a pseudo-polynomial-time algorithm for this problem, which suggests that the problem cannot be strongly NP-hard unless P = NP. When the processing times and due dates are agreeable, Emmons [5] showed that the problem can be solved optimally by ordering the jobs in non-decreasing order of their processing times. Karp [7] proved that the problem of minimizing weighted number of tardy jobs on a single machine is NP-hard even if all the jobs are subject to the same due date. Lawler and Moore [9] presented a pseudo-polynomial-time algorithm for this problem.

* Corresponding author. Tel.: +852 27667364; fax: +852 23302704.
E-mail address: lgtctng@polyu.edu.hk (C.T. Ng).

0304-3975/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.12.039

Ikura and Gimple [6] may be the first researchers to study the batch scheduling problems. Many results in this area have since been obtained. Lee et al. [10] provided polynomial-time algorithms for the problems of scheduling jobs with agreeable processing times and due dates on a single batch processing machine to minimize maximum tardiness and number of tardy jobs. Li and Lee [11] proved that the batch scheduling problem with agreeable release dates and deadlines is strongly NP-hard. They developed polynomial-time algorithms for the problems of minimizing maximum tardiness and number of tardy jobs when all the jobs have agreeable release dates, due dates and processing times. Brucker et al. [1] provided an extensive discussion of the problems minimizing various regular objectives on an unbounded batch processing machine and on a bounded batch processing machine. For the scheduling problems on a batch processing machine with unbounded capacity, they developed polynomial-time dynamic programming algorithms for several objectives and proved that the problems of minimizing weighted number of tardy jobs and total weighted tardiness are ordinary NP-hard. For the scheduling problems on a batch processing machine with bounded capacity, they presented a dynamic programming algorithm for the problem with the total completion time objective and proved that the problems with due date related objectives are strongly NP-hard. Cheng et al. [3] proved that scheduling jobs with release dates and deadlines on an unbounded batch processing machine is NP-hard even if all the jobs are subject to agreeable processing times and deadlines. They developed polynomial-time algorithms for several special cases. Liu et al. [13] showed that the problem of minimizing total tardiness on a single unbounded batch processing machine is NP-hard in the ordinary sense and provided a pseudo-polynomial-time algorithm for the problems with release dates and several regular objectives on a single unbounded batch processing machine.

Given the strong NP-hardness of the batch scheduling problems with due date related objectives, we study in this paper the scheduling problems with agreeable processing times and due dates to minimize total tardiness, and weighted number of tardy jobs. These problems are interesting since in reality the job with a longer processing time is often assigned a larger due date [2,11], then the jobs satisfy the condition of agreeable processing times and due dates.

The rest of this paper is organized as follows. In Section 2 we discuss the assumptions and notation that will be used in this paper. In Section 3 we prove that the problem of scheduling jobs with agreeable processing times and due dates on a batch processing machine to minimize total tardiness is NP-hard even if the machine can process at most two jobs at the same time. We present a pseudo-polynomial-time algorithm for an NP-hard special case of this problem. In Section 4 we develop a pseudo-polynomial-time algorithm for the NP-hard problem of scheduling jobs with agreeable processing times and due dates on a single batch processing machine to minimize weighted number of tardy jobs, which suggests that this problem cannot be strongly NP-hard unless P = NP. Finally, we draw some conclusions and suggest some future research directions in Section 5.

2. Assumptions and notations

In this paper we make assumptions and use notation about the jobs and the machine as follows:

- There are n jobs, all of which are available at time zero, to be processed on a single batch processing machine. The processing time of job J_j is denoted by p_j, its weight by w_j, and its due date by d_j. We say that the processing times and due dates are agreeable if $p_i < p_j$ implies that $d_i \leq d_j$ $(1 \leq i < j \leq n)$. Since we study the problems with agreeable processing times and due dates, throughout this paper we assume that all the jobs have been indexed in non-decreasing order of their processing times and due dates such that $p_1 \leq p_2 \leq \cdots \leq p_n$ and $d_1 \leq d_2 \leq \cdots \leq d_n$.
- The batch processing machine can process up to B jobs at the same time. A batch is called *full* if it contains exactly B jobs; otherwise, it is called a *partial* batch. The processing time of batch B_i is denoted by $p(B_i)$, which is equal to the longest processing time of the jobs in the batch. The number of jobs in batch B_i is denoted by $|B_i|$.

We use the common three-field notation to denote the scheduling problems under study. For example, $1|B, \text{agr}(p_j, d_j)|\sum T_j$ denotes the problem of scheduling jobs with agreeable processing times and due dates on a single bounded batch processing machine to minimize total tardiness.

3. Minimizing total tardiness

The problem $1|\text{agr}(p_j, d_j)|\sum T_j$ can be solved in polynomial time [5]. Brucker et al. [1] showed that the problem $1|B = 2|L_{\max} \leq 0$ is strongly NP-hard, which indicates that the problem $1|B = 2|\sum T_j$ is strongly NP-hard, too. Liu and Yu [12] proved that the problem $1|B = 2, r_j \in \{0, r\}|C_{\max}$ is NP-hard in the ordinary sense. By scheduling the jobs in a backward way, the problem $1|B = 2, r_j \in \{0, r\}|C_{\max}$ can be transformed into the problem $1|B = 2, d_j \in \{d_1, d_2\}|L_{\max} \leq 0$, where d_1 and d_2 are two distinct due dates and $0 < d_1 < d_2$. It follows that the problem $1|B = 2, d_j \in \{d_1, d_2\}| \sum T_j$ is NP-hard, too. In this section we prove that the problem $1|B, \mathrm{agr}(p_j, d_j)| \sum T_j$ is NP-hard even if $B = 2$ by reducing the partition problem to it. Our reduction is a modification of the one in [13]. We first introduce a structural property of a class of optimal schedules for the case of $B = 2$.

Lemma 1. *There exists an optimal schedule for the problem $1|B = 2, \mathrm{agr}(p_j, d_j)| \sum T_j$ such that all the batches contain consecutively indexed job(s).*

Proof. Consider an optimal schedule in which jobs J_i and J_j are processed in the same batch B_l and job J_s $(i < s < j)$ is processed in another batch B_k. Let C_1 and C_2 denote the completion times of batches B_l and B_k, respectively.

If B_l is processed before B_k, exchange job J_s and job J_j by moving J_s to B_l and J_j to B_k. Since $p_i \leq p_s \leq p_j$, the completion times of all the batches do not increase. Denoting the new completion times of batches B_l and B_k as C_1' and C_2', respectively, we have $C_1' \leq C_1$ and $C_2' \leq C_2$. Let T_s, T_j and T_s', T_j' denote the tardiness of jobs J_s and J_j before and after the job exchange, respectively. Therefore:

$$T_j = \max\{0, C_1 - d_j\}, \qquad T_s = \max\{0, C_2 - d_s\},$$
$$T_s' = \max\{0, C_1' - d_s\}, \qquad T_j' = \max\{0, C_2' - d_j\}.$$

Because $C_1' \leq C_1$ and $C_2' \leq C_2$, it follows that:

$$\begin{aligned} &T_s + T_j - (T_s' + T_j') \\ &\quad \geq \max\{0, C_1 - d_j\} + \max\{0, C_2 - d_s\} - (\max\{0, C_1 - d_s\} + \max\{0, C_2 - d_j\}). \end{aligned}$$

Since $d_s \leq d_j$, $C_1 < C_2$, we know that both $C_1 - d_s$ and $C_2 - d_j$ lie within the time interval $(C_1 - d_j, C_2 - d_s)$. Moreover, $C_1 - d_s - (C_1 - d_j) = C_2 - d_s - (C_2 - d_j) = d_j - d_s$ and the non-negative increasing function $x^+ = \max\{0, x\}$ is convex, and we obtain:

$$\begin{aligned} &T_s + T_j - (T_s' + T_j') \\ &\qquad \geq (C_1 - d_j)^+ + (C_2 - d_s)^+ - ((C_1 - d_s)^+ + (C_2 - d_j)^+) \\ &\qquad = (C_2 - d_s)^+ - (C_2 - d_j)^+ - ((C_1 - d_s)^+ - (C_1 - d_j)^+) \\ &\qquad \geq 0. \end{aligned}$$

Thus, the total tardiness does not increase after the job interchange.

If B_l is processed after B_k, exchange job J_i and job J_s by moving J_s to B_l and J_i to B_k. Repeat the above arguments; the total tardiness does not increase after the job interchange either.

Repeating the above procedures, we can construct an optimal schedule with all the batches containing consecutively indexed jobs. □

3.1. NP-hardness proof

Partition: Given t positive integers $\{a_1, \ldots, a_t\}$ such that $\sum a_j = 2A$, is there a partition of the index set $\{1, \ldots, t\}$ into A_1, A_2 such that $\sum_{j \in A_1} a_j = \sum_{j \in A_2} a_j = A$?

Given any instance of the partition problem, we first define the following $3t + 1$ integers:

$$M_t = \sum_{i=1}^{t} (t - i) a_i + 6A,$$

$$M_k = 3 \sum_{i=k+1}^{t} M_i + \sum_{i=1}^{t} (t - i) a_i + 6A, \quad k = t - 1, \ldots, 1,$$

$$L_1 = 11 \sum_{i=1}^{t} M_i + \sum_{i=1}^{t} (t - i) a_i + 4A,$$

$$L_k = 2 \sum_{i=1}^{k-1} L_i + 11 \sum_{i=1}^{t} M_i + \sum_{i=1}^{t} (t - i) a_i + 4A, \quad k = 2, \ldots, 2t + 1.$$

Obviously, the above defined $3t + 1$ integers satisfy:

$$2A \ll M_t \ll M_{t-1} \ll \cdots \ll M_1 \ll L_1 \ll L_2 \ll \cdots \ll L_{2t+1}.$$

We now construct a scheduling instance with $n = 6t + 2$ jobs and $B = 2$. For each k ($k = 1, \ldots, 2t$), define three type k jobs J_k^i ($i = 1, 2, 3$). For $k = 1, \ldots, t$; $i = 1, 2, 3$, job J_{2k-1}^i has the following processing time p_{2k-1}^i and due date d_{2k-1}^i.

$$\begin{aligned}
p_{2k-1}^1 &= L_{2k-1}, \\
p_{2k-1}^2 &= L_{2k-1} + 2M_k, \\
p_{2k-1}^3 &= L_{2k-1} + 3M_k, \\
d_{2k-1}^1 &= 2\sum_{i=1}^{2k-2} L_i + 8\sum_{i=1}^{k-1} M_i + L_{2k-1} + 2M_k + 2A, \\
d_{2k-1}^2 &= 2\sum_{i=1}^{2k-1} L_i + 8\sum_{i=1}^{k-1} M_i, \\
d_{2k-1}^3 &= 2\sum_{i=1}^{2k-1} L_i + 8\sum_{i=1}^{k-1} M_i + 5M_k + 2A.
\end{aligned}$$

For $k = 1, \ldots, t;\ i = 1, 2, 3$, job J_{2k}^i has the following processing time p_{2k}^i and due date d_{2k}^i.

$$\begin{aligned}
p_{2k}^1 &= L_{2k}, \\
p_{2k}^2 &= L_{2k} + 2M_k + a_k, \\
p_{2k}^3 &= L_{2k} + 3M_k, \\
d_{2k}^1 &= 2\sum_{i=1}^{2k-1} L_i + 8\sum_{i=1}^{k-1} M_i + L_{2k} + 5M_k + 2A, \\
d_{2k}^2 &= 2\sum_{i=1}^{2k} L_i + 8\sum_{i=1}^{k-1} M_i + 5M_k - (t - k + 1)a_k, \\
d_{2k}^3 &= 2\sum_{i=1}^{2k} L_i + 8\sum_{i=1}^{k} M_i + 2A.
\end{aligned}$$

In addition, we define two type $2t+1$ jobs J_{2t+1}^1 and J_{2t+1}^2 with the same processing time $p_{2t+1}^1 = p_{2t+1}^2 = L_{2t+1}$ and the same due date $d_{2t+1}^1 = d_{2t+1}^2 = 2\sum_{i=1}^{2t} L_i + L_{2t+1} + 8\sum_{i=1}^{t} M_i + A$. According to the definition of processing times and due dates, we know that:

$$\begin{aligned}
&p_1^1 < p_1^2 < p_1^3 < \cdots < p_k^1 < p_k^2 < p_k^3 < \cdots < p_{2t}^1 < p_{2t}^2 < p_{2t}^3 < p_{2t+1}^1 = p_{2t+1}^2, \\
&d_1^1 < d_1^2 < d_1^3 < \cdots < d_k^1 < d_k^2 < d_k^3 < \cdots < d_{2t}^1 < d_{2t}^2 < d_{2t}^3 < d_{2t+1}^1 = d_{2t+1}^2.
\end{aligned}$$

Set the threshold value of the total tardiness as

$$T^* = 3\sum_{i=1}^{t} M_i + \sum_{i=1}^{t} (t - i)a_i + A.$$

We call a schedule satisfying $\sum T_j \le T^*$ a *feasible schedule*. According to Lemma 1, any feasible schedule can be transformed into one with each batch containing consecutively indexed jobs; we call this schedule a *feasible schedule with consecutively indexed jobs*, and then we know that there exists a feasible schedule for the constructed scheduling instance if and only if there exists a feasible schedule with consecutively indexed jobs for it. We will prove that there exists a feasible schedule with consecutively indexed jobs for the scheduling instance if and only if the partition instance has a solution.

Obviously, the reduction for constructing the scheduling instance is polynomial under binary encoding. We next explore the properties of any feasible schedule with consecutively indexed jobs. For a given schedule, $k = 1, \ldots, 2t;\ i = 1, 2, 3$ and $k = 2t + 1;\ i = 1, 2$, let T_k^i denote the tardiness of job J_k^i.

Lemma 2. *In any feasible schedule with consecutively indexed jobs, each batch contains only jobs of one type; the three jobs of the same type must be assigned to two batches; and all the batches are processed in non-decreasing order of their processing times.*

Proof. Consider a schedule in which a type k job and a type h $(1 \le k < h \le 2t + 1)$ job are processed in the same batch. The tardiness of the type k job in this batch is at least

$$\begin{aligned} L_h - d_k^3 &\ge L_h - \left(2\sum_{i=1}^{k} L_i + 8\sum_{i=1}^{t} M_i + 2A\right) \ge L_{k+1} - \left(2\sum_{i=1}^{k} L_i + 8\sum_{i=1}^{t} M_i + 2A\right) \\ &> 3\sum_{i=1}^{t} M_i + \sum_{i=1}^{t}(t-i)a_i + A = T^*. \end{aligned}$$

Therefore, in any feasible schedule, each batch contains only jobs of the same type.

Consider a schedule in which the three jobs of type k $(1 \le k \le 2t)$ are assigned to three batches, and the tardiness of the last type k job is at least:

$$\begin{aligned} 3L_k - d_k^3 &\ge 3L_k - \left(2\sum_{i=1}^{k} L_i + 8\sum_{i=1}^{t} M_i + 2A\right) = L_k - \left(2\sum_{i=1}^{k-1} L_i + 8\sum_{i=1}^{t} M_i + 2A\right) \\ &> 3\sum_{i=1}^{t} M_i + \sum_{i=1}^{t}(t-i)a_i + A = T^*. \end{aligned}$$

Since the machine can process at most two jobs at the same time, the three jobs J_k^i $(i = 1, 2, 3)$ of the same type k $(k = 1, \ldots, 2t)$ must be assigned to two batches: $\{J_k^1, J_k^2\}$ and $\{J_k^3\}$; or $\{J_k^1\}$ and $\{J_k^2, J_k^3\}$. In the same way, we know that the two jobs J_{2t+1}^1 and J_{2t+1}^2 must be assigned to the same batch.

Consider a schedule in which batch B_k contains one type k job, batch B_h contains one type h $(1 \le k < h \le 2t+1)$ job and B_h is processed before B_k. Then the tardiness of the type k job in B_k is at least:

$$\begin{aligned} L_h + L_k - d_k^3 &\ge L_{k+1} + L_k - \left(2\sum_{i=1}^{k} L_i + 8\sum_{i=1}^{t} M_i + 2A\right) \\ &> 3\sum_{i=1}^{t} M_i + \sum_{i=1}^{t}(t-i)a_i + A = T^*. \end{aligned}$$

Consider a schedule in which two batches containing type k $(1 \le k \le 2t)$ jobs satisfy that the batch containing job J_k^1 is processed after the batch containing job J_k^3, and we obtain:

$$\begin{aligned} T_k^1 > L_k + L_k - d_k^1 &> 2L_k - \left(2\sum_{i=1}^{k-1} L_i + L_k + 8\sum_{i=1}^{t} M_i + 2A\right) \\ &= L_k - \left(2\sum_{i=1}^{k-1} L_i + 8\sum_{i=1}^{t} M_i + 2A\right) \\ &> 3\sum_{i=1}^{t} M_i + \sum_{i=1}^{t}(t-i)a_i + A = T^*. \end{aligned}$$

Hence, in any feasible schedule, all the batches are processed in non-decreasing order of their processing times. □

Lemma 3. *In any feasible schedule with consecutively indexed jobs, for each k $(k = 1, \ldots, t)$, the six jobs of type $2k - 1$ and $2k$ are assigned to four batches: $\{J_{2k-1}^1, J_{2k-1}^2\}$, $\{J_{2k-1}^3\}$, $\{J_{2k}^1\}$, $\{J_{2k}^2, J_{2k}^3\}$; or $\{J_{2k-1}^1\}$, $\{J_{2k-1}^2, J_{2k-1}^3\}$,*

$\{J_{2k}^1, J_{2k}^2\}$, $\{J_{2k}^3\}$. *The first pattern is called* 2112 *with total processing time* $2(L_{2k-1} + L_{2k}) + 8M_k$ *and the second pattern is called* 1221 *with total processing time* $2(L_{2k-1} + L_{2k}) + 8M_k + a_k$.

Proof. We prove this lemma by induction on k. For the six type 1 and type 2 jobs, if the two batches $\{J_1^1, J_1^2\}$ and $\{J_2^1, J_2^2\}$ exist, then the total tardiness of J_2^1 and J_2^3 is

$$\begin{aligned} p_1^2 + p_1^3 + p_2^2 - d_2^1 + p_1^2 + p_1^3 + p_2^2 + p_2^3 - d_2^3 &> 4M_1 - 4A \\ &> 3\sum_{i=1}^{t} M_i + \sum_{i=1}^{t}(t-i)a_i + A = T^*. \end{aligned}$$

If the two batches $\{J_1^2, J_1^3\}$ and $\{J_2^2, J_2^3\}$ exist, then the total tardiness of J_1^2 and J_2^2 is

$$\begin{aligned} p_1^1 + p_1^3 - d_1^2 + p_1^1 + p_1^3 + p_2^1 + p_2^3 - d_2^2 &> 4M_1 \\ &> 3\sum_{i=1}^{t} M_i + \sum_{i=1}^{t}(t-i)a_i + A = T^*. \end{aligned}$$

Thus, in any feasible schedule with consecutively indexed jobs the six type 1 and type 2 jobs can only be assigned to four batches: $\{J_1^1, J_1^2\}$, $\{J_1^3\}$, $\{J_2^1\}$, $\{J_2^2, J_2^3\}$; or $\{J_1^1\}$, $\{J_1^2, J_1^3\}$, $\{J_2^1, J_2^2\}$, $\{J_2^3\}$.

Suppose that there exists a feasible schedule with consecutively indexed jobs such that this lemma is true for $i = 1, \ldots, k-1$, but it does not hold for k. Then the total processing time of the $(4i-3)th$, $(4i-2)th$, $(4i-1)th$, and $(4i)th$ batches is at least $2(L_{2i-1} + L_{2i}) + 8M_i$. By computation, if the six jobs of type $2i-1$ and $2i$ are of Pattern 1221, then the tardiness of job J_{2i-1}^2 is at least $3M_i$; if the six jobs of type $2i-1$ and $2i$ are of Pattern 2112, then the tardiness of job J_{2i}^2 is also at least $3M_i$. Hence, the total tardiness of the first $4(k-1)$ batches is at least $3\sum_{i=1}^{k-1} M_i$. For the six jobs of type $2k-1$ and $2k$, we can discuss this case similar to the case of type 1 and type 2 jobs. It follows that the total tardiness will be larger than T^* and this is a contradiction to the feasible schedule assumption. □

Let A_1 be the index set of k $(k = 1, \ldots, t)$ such that the six jobs of type $2k-1$ and $2k$ are of Pattern 2112 and $A_2 = \{1, \ldots, t\} - A_1$. According to the above discussion, if $k \in A_1$, then the total processing time of the four batches $\{J_{2k-1}^1, J_{2k-1}^2\}$, $\{J_{2k-1}^3\}$, $\{J_{2k}^1\}$, and $\{J_{2k}^2, J_{2k}^3\}$ is $2(L_{2k-1} + L_{2k}) + 8M_k$, and if $k \in A_2$, then the total processing time of the four bathes $\{J_{2k-1}^1\}$, $\{J_{2k-1}^2, J_{2k-1}^3\}$, $\{J_{2k}^1, J_{2k}^2\}$, and $\{J_{2k}^3\}$ is $2(L_{2k-1} + L_{2k}) + 8M_k + a_k$. Therefore, if $k \in A_1$ $(k = 1, \ldots, t)$, then J_{2k}^2 is the only tardy job of the four batches and its tardiness is

$$3M_k + (t-k+1)a_k + \sum\{a_i | i < k, i \in A_2\}.$$

If $k \in A_2$ $(k = 1, \ldots, t)$, then J_{2k-1}^2 is the only tardy job of the four batches and its tardiness is

$$3M_k + \sum\{a_i | i < k, i \in A_2\}.$$

Combining all the properties we have derived for any feasible schedule with consecutively indexed jobs, we obtain the following conclusion.

Lemma 4. *There exists a feasible schedule with consecutively indexed jobs for the scheduling instance such that* $\sum T_j \le T^*$ *if and only if* $\sum_{i\in A_1} a_i = \sum_{i\in A_2} a_i = A$ *holds for the partition instance.*

Proof. From the above discussion, the total tardiness is equal to:

$$\sum T_j = \sum_{k=1}^{t}\left(3M_k + \sum\{a_i | i < k, i \in A_2\}\right) + \sum_{k\in A_1}(t-k+1)a_k + 2\max\left\{0, \sum_{k\in A_2} a_k - A\right\},$$

where the third term on the right-hand side is the total tardiness of jobs J_{2t+1}^1 and J_{2t+1}^2. Since:

$$\sum_{k=1}^{t}\sum\{a_i | i < k, i \in A_2\} = \sum_{i\in A_2}(t-i)a_i,$$

we obtain:

$$\sum T_j = 3\sum_{k=1}^{t} M_k + \sum_{k=1}^{t}(t-k)a_k + \sum_{k\in A_1} a_k + 2\max\left\{0, \sum_{k\in A_2} a_k - A\right\}.$$

Therefore, there exists a feasible schedule with consecutively indexed jobs such that $\sum T_j \le T^*$ if and only if $\sum_{k\in A_1} a_k + 2\max\{0, \sum_{k\in A_2} a_k - A\} \le A$. Hence, $\sum_{k\in A_1} a_k \le A$ and $\sum_{k\in A_2} a_k \le A$. Because $\sum_{k\in A_1} a_k + \sum_{k\in A_2} a_k = 2A$, we get $\sum_{k\in A_1} a_k = \sum_{k\in A_2} a_k = A$. $\square$

Based on the above lemmas, we obtain the following theorem:

Theorem 1. *The problem* $1|B, \mathrm{agr}(p_j, d_j)|\sum T_j$ *is NP-hard even if* $B = 2$.

3.2. Pseudo-polynomial algorithm

According to the construction of the scheduling instance from a partition instance in the proof of the NP-hardness of the problem $1|B, \mathrm{agr}(p_j, d_j)|\sum T_j$ with $B = 2$, we know that the constructed scheduling instance satisfies $p_{i+1} - p_i \le d_{i+1} - d_i$ $(i = 1, \ldots, n-1)$. Therefore, the problem $1|B, \mathrm{agr}(p_j, d_j)|\sum T_j$ is NP-hard even if $B = 2$ and $p_{i+1} - p_i \le d_{i+1} - d_i$ $(i = 1, \ldots, n-1)$. We next develop a pseudo-polynomial-time dynamic programming algorithm for this problem. Before presenting the dynamic programming algorithm, we first provide some structural properties of a class of optimal schedules. We call batch B_j a *deferred* batch of B_k (or batch B_j is deferred by B_k) if B_j is scheduled after B_k and $p(B_k) > p(B_j)$, and batch B_k is called a *deferring* batch of B_j.

Lemma 5. *For the problem* $1|B, \mathrm{agr}(p_j, d_j)|\sum T_j$*, there exists an optimal schedule in which all the deferring batches are full, all the deferred batches are partial and all the jobs in the deferred batches are tardy.*

Proof. Consider an optimal schedule containing a deferring partial batch. Move jobs from its deferred batches to this deferring partial batch until it becomes a full batch or it is still a partial batch but all the batches after it have processing times longer than the processing time of this batch (in this case, it is not a deferring batch any more). The completion times of the moved jobs decrease and the completion times of the other jobs do not increase, thus the schedule remains optimal. Repeating the above procedure, we obtain an optimal schedule with all the deferring batches being full.

Consider an optimal schedule containing a full deferred batch B_i. From the above discussion, we can assume that its deferring batches are full. Denote one of its deferring batches as B_k and the completion times of batches B_k and B_i as C_1 and C_2, respectively. Order the $2B$ jobs in batches B_k and B_i in increasing order of their indices and assign the B jobs with the smallest indices to batch B_k and the remaining B jobs to batch B_i. After the rearrangement, batch B_i is not deferred by batch B_k any more. Denote the new completion times of batches B_k and B_i as C_1' and C_2', respectively. We have $C_1' \le C_1$ and $C_2' \le C_2$ and the completion times of the other batches do not increase. The tardiness of the jobs that were originally in B_k (B_i) and are now still in B_k (B_i) does not increase. If there is a job J_k that was originally in batch B_k and is now in batch B_i, then there must exist another job J_i with $p_k \ge p_i$ and $d_k \ge d_i$ that was originally in batch B_i and is now in batch B_k. Therefore, the decreased tardiness by exchanging jobs J_k and J_i is

$$\begin{aligned}
&\max\{0, C_1 - d_k\} + \max\{0, C_2 - d_i\} - \max\{0, C_1' - d_i\} - \max\{0, C_2' - d_k\} \\
&\ge \max\{0, C_1 - d_k\} + \max\{0, C_2 - d_i\} - \max\{0, C_1 - d_i\} - \max\{0, C_2 - d_k\} \\
&= (C_2 - d_i)^+ - (C_2 - d_k)^+ - ((C_1 - d_i)^+ - (C_1 - d_k)^+).
\end{aligned}$$

Following the proof of Lemma 1, we know that the decreased tardiness is non-negative. Repeating the above discussions, we obtain an optimal schedule with all the deferred batches being partial.

Consider an optimal schedule in which deferred batch B_i contains at least one job finished on time. Since all the jobs have agreeable processing times and due dates, in batch B_i the job with the largest index must be on time. Denote this job as J_i and one of the deferring batches of B_i as B_k. Denote the completion times of batches B_k and B_i as C_1 and C_2, respectively. Denote the number of jobs in batch B_k with indices larger than that of job J_i as a. We have $a \le B$. If $a + |B_i| \le B$, move the a jobs in batch B_k with indices larger than that of J_i to B_i; otherwise, move $a + |B_i| - B$ jobs in batch B_i with the smallest indices to B_k and the a jobs in batch B_k with indices larger than that of J_i to B_i. The completion times of the other batches do not increase, the jobs moved from B_k to B_i are on time, the completion times of jobs moved from B_i to B_k are decreased, and the completion times of the other jobs in these two batches do not increase. Therefore, the total tardiness does not increase and batch B_i is not a deferred batch of B_k any more. Repeat the above procedures for an optimal schedule in which all the jobs in the deferred batches are tardy. □

From Lemma 5, we obtain the following lemma, which establishes that the partial batches and the full batches are arranged in non-decreasing order of their processing times.

Lemma 6. *For the problem* $1|B, \mathrm{agr}(p_j, d_j)|\sum T_j$, *there exists an optimal schedule in which all the partial bathes and all the full batches are arranged in non-decreasing order of their processing times.*

Lemma 7. *For the problem* $1|B, \mathrm{agr}(p_j, d_j)|\sum T_j$ *with* $B = 2$ *and* $p_{i+1} - p_i \le d_{i+1} - d_i$ $(i = 1, \ldots, n-1)$, *if batch* $\{J_i\}$ *is deferred by batch* $\{J_j, J_k\}$ *in any optimal schedule that satisfies Lemmas 1, 5 and 6, we have* $\frac{p_k}{2} \le p_i$ $(1 \le i < j < k \le n)$.

Proof. According to Lemma 1 and the definition of deferred batch, we obtain that $p_i \le p_j \le p_k$, $d_i \le d_j \le d_k$ and $p_i < p_k$. Without loss of generality, we assume that batch $\{J_i\}$ is processed immediately after batch $\{J_j, J_k\}$; otherwise, following Lemmas 5 and 6, we can find another deferred partial batch and(or) another deferring full batch such that the partial batch is processed immediately after the full batch and the job in the partial batch is tardy. Suppose that $\frac{p_k}{2} > p_i$ in an optimal schedule σ. Denote the starting time of batch $\{J_j, J_k\}$ as C in σ. We construct another schedule σ' by exchanging batches $\{J_i\}$ and $\{J_j, J_k\}$ in σ. Then job J_i must be on time in σ'; otherwise, from σ to σ', the decreased tardiness of job J_i is p_k and the increased total tardiness of jobs J_j and J_k is at most $2p_i$. Since $\frac{p_k}{2} > p_i$, it follows that schedule σ' is better than σ and this is a contradiction. We obtain $C + p_i \le d_i$ and $C + p_k - d_k \le d_i - p_i + p_k - d_k = p_k - p_i - (d_k - d_i) \le 0$. Hence, job J_k is on time in schedule σ. We consider five possible cases.

Case 1. Job J_j is tardy in σ and job J_k is on time in σ', then job J_j must be tardy in σ'. The decreased total tardiness from σ to σ' is

$$C + p_k - d_j + C + p_k + p_i - d_i - (C + p_i + p_k - d_j) = C + p_k - d_i.$$

Since job J_j is tardy in σ, we have $C + p_k - d_j > 0$. Hence, $C + p_k - d_i > d_j - d_i \ge 0$.

Case 2. Job J_j is tardy in σ and job J_k is tardy in σ', then job J_j must be tardy in σ'. The decreased total tardiness from σ to σ' is:

$$C + p_k - d_j + C + p_k + p_i - d_i - (C + p_i + p_k - d_j + C + p_i + p_k - d_k) = d_k - d_i - p_i.$$

Since $d_k - d_i \ge p_k - p_i$, it follows that $d_k - d_i - p_i \ge p_k - 2p_i > 0$.

Case 3. Job J_j is on time in both σ and σ', so job J_k must be on time in σ' since $d_k \ge d_j$. Obviously, the decreased total tardiness from σ to σ' is strictly larger than zero since job J_i is tardy in σ according to Lemma 5.

Case 4. Job J_j is on time in σ, job J_j is tardy in σ' and job J_k is on time in σ'. The decreased total tardiness from σ to σ' is:

$$C + p_k + p_i - d_i - (C + p_i + p_k - d_j) = d_j - d_i.$$

If $d_j > d_i$, we obtain $d_j - d_i > 0$. The decreased total tardiness is positive. If $d_j = d_i$, we have $0 = d_j - d_i \ge p_j - p_i \ge 0$, thus $p_j = p_i$. We construct another schedule σ'' by assigning jobs J_i and J_j to one batch and this batch is immediately followed by a batch containing only job J_k. Then jobs J_i, J_j, J_k are on time in σ'' while job J_i is tardy in σ. It follows that the decreased total tardiness is strictly larger than zero from σ to σ''.

Case 5. Job J_j is on time in σ, and jobs J_j and J_k are tardy in σ'. The decreased total tardiness from σ to σ' is

$$C + p_k + p_i - d_i - (C + p_i + p_k - d_j + C + p_i + p_k - d_k) = d_k - d_i - p_i + d_j - C - p_k.$$

Since $d_k - d_i \ge p_k - p_i$, it follows that $d_k - d_i - p_i \ge p_k - 2p_i > 0$. Since job J_j is on time in σ, we have $C + p_k \le d_j$. Hence, the decreased total tardiness is strictly larger than zero.

We have considered all the possible cases and for each case we can construct a better schedule than schedule σ, which implies that σ is not optimal. □

Lemma 8. *For the problem* $1|B, \mathrm{agr}(p_j, d_j)|\sum T_j$ *with* $B = 2$ *and* $p_{i+1} - p_i \le d_{i+1} - d_i$ $(i = 1, \ldots, n-1)$, *the number of deferred batches of any full batch is at most one in any optimal schedule that satisfies Lemmas 1 and 5–7.*

Proof. Suppose there are more than one deferred partial batches of full batch B_k in an optimal schedule σ. Denote the nearest two deferred batches from batch B_k as $\{J_i\}$ and $\{J_j\}$, respectively. In accordance with Lemma 7, we have $\frac{p(B_k)}{2} \leq p_i$ and $\frac{p(B_k)}{2} \leq p_j$. Following Lemmas 5 and 6, we know that jobs J_i and J_j are tardy in σ and $p_i < p_j$. Then σ can be represented as the following form:

$$\sigma = S_0 B_k \{J_i\} S_1 \{J_j\} S_2,$$

where S_0, S_1, S_2 are three blocks of batches and S_1 contains full batches with processing times longer than or equal to $p(B_k)$ according to Lemmas 5 and 6.

We construct another schedule σ' from σ by assigning job J_i and J_j to the same batch and scheduling it immediately after batch B_k. Hence,

$$\sigma' = S_0 B_k \{J_i, J_j\} S_1 S_2.$$

If job J_j is on time in σ', then exchange batches B_k and $\{J_i\}$ in σ. The jobs in B_k are on time while the tardiness of J_i is decreased, a contradiction to the optimality of σ. Therefore, both jobs J_i and J_j are tardy in σ'. From σ to σ', the tardiness of job J_i increases by $p_j - p_i$, the total tardiness of jobs in S_1 increases by at most $(p_j - p_i)|S_1|$, and the tardiness of job J_j decreases by $p_i + p(S_1)$, where $p(S_1)$ and $|S_1|$ denote the total processing time of batches in S_1 and the total number of jobs in S_1. Therefore, the total tardiness is decreased by at least:

$$p_i + p(S_1) - (p_j - p_i) - (p_j - p_i)|S_1| = 2p_i - p_j + p(S_1) - (p_j - p_i)|S_1|.$$

Since $\frac{p(B_k)}{2} \leq p_i$, $p(B_k) > p_j$ and $\frac{p(B_k)}{2} \leq \frac{p(S_1)}{|S_1|}$, it follows that $2p_i - p_j > 0$, $p_j - p_i < \frac{p(B_k)}{2} \leq \frac{p(S_1)}{|S_1|}$ and $p(S_1) - (p_j - p_i)|S_1| > 0$. Hence, the decreased total tardiness is positive and σ is not an optimal schedule. $\square$

We now present a dynamic programming algorithm for the NP-hard problem $1|B, \text{agr}(p_j, d_j)| \sum T_j$ with $B = 2$ and $p_{i+1} - p_i \leq d_{i+1} - d_i$ $(i = 1, \ldots, n-1)$. According to Lemma 8, the number of deferred batch of any full batch does not exceed one. Let $f(C, j, i)$ denote the minimum total tardiness when jobs $J_1, \ldots, J_{i-1}$ and $J_{i+1}, \ldots, J_j$ are scheduled such that the unscheduled job J_i is deferred by the full batch containing job J_j and the completion time of the last batch is C. If no unscheduled job is deferred by the batch containing job J_j, the minimum total tardiness is denoted as $f(C, j, o)$.

Let $f(0, 0, o) = 0$. For $C = p_1, \ldots, \sum p_j$, $j = 1, \ldots, n$, and $i < j$, the value $f(C, j, o)$ can be obtained by the following recurrence equations:

$$f(C, j, o) = \min \begin{cases} \min\limits_{i<j}\{f(C - p_i, j, i) + \max\{0, C - d_i\}\}, \\ \min\limits_{1 \leq l \leq 2} \left\{ f(C - p_j, j - l, o) + \sum\limits_{k=1}^{l} \max\{0, C - d_{j-k+1}\} \right\}, \end{cases}$$

where the first term is taken by adding the deferred batch of the batch containing job J_j and the second term is taken by adding a full or partial batch containing job J_j that does not defer any batch.

The value $f(C, j, i)$ can be obtained by the following recurrence equations:

$$f(C, j, i) = \min \begin{cases} f(C - p_j, j - 2, i) + \sum\limits_{k=1}^{2} \max\{0, C - d_{j-k+1}\}, & \text{if } i < j - 2, \\ f(C - p_j, j - 3, o) + \sum\limits_{k=1}^{2} \max\{0, C - d_{j-k+1}\}, & \text{if } i = j - 2, \end{cases}$$

where the two terms are both taken by adding the full batch containing job J_j while the first term corresponds to job J_i being deferred by the batch containing job J_{j-2} and the second term corresponds to job J_i being job J_{j-2}.

The optimal value is equal to $\min_{p_1 \leq C \leq \sum_{k=1}^{n} p_k} f(C, n, o)$ and the time complexity of this dynamic programming algorithm is $O(n^3 \sum_{k=1}^{n} p_k)$, which is a pseudo-polynomial-time algorithm.

4. Minimizing the weighted number of tardy job

We know that the problem $1||\sum w_j U_j$ is ordinary NP-hard even if all the jobs have the same due dates. Thus, the problem $1|B, \text{agr}(p_j, d_j)| \sum w_j U_j$ is NP-hard, too. In this section we develop a pseudo-polynomial-time algorithm for this problem, and therefore show that the problem cannot be strongly NP-hard unless P = NP. According to Lee et al. [10], we have the following conclusion.

Lemma 9. *There exists an optimal schedule for the problem* $1|B, \mathrm{agr}(p_j, d_j)|\sum w_j U_j$ *in which all the on-time jobs are processed before all the tardy jobs and the on-time jobs are arranged in non-decreasing order of their indices.*

We now present a dynamic programming algorithm that runs in pseudo-polynomial time. Let $F_j(k, l, t)$ denote the minimum weighted number of tardy jobs when jobs $J_1, \dots, J_j$ are scheduled, the last on time batch can be expanded to comprise job J_k and $l \le B$ is the number of jobs in the last on time batch, and the completion time of the last on time batch is t. We develop the following dynamic programming algorithm.

$$F_0(k, l, t) = \begin{cases} 0, & \text{if } k = t = 0,\ l = B, \\ \infty, & \text{otherwise.} \end{cases}$$

For $j = 1, \dots, n$, $k = 0, \dots, n$, $l = 1, \dots, B$, and $t = 0, \dots, \sum_{j=1}^{n} p_j$, the recursion equations are

$$F_j(k, l, t) = \begin{cases} \min \begin{cases} F_{j-1}(k, l-1, t), & \text{if } t \le d_j, \\ \min\limits_{h \le j-1, 1 \le l' \le B} \{F_{j-1}(h, l', t - p_k)\}, & \text{if } l = 1, t \le d_j, \\ F_{j-1}(k, l, t) + w_j, & \text{if } t \le d_j, \end{cases} \\ F_{j-1}(k, l, t) + w_j, & \text{if } t > d_j. \end{cases}$$

The optimal value is equal to $\min_{0 \le k \le n, 1 \le l \le B, 0 \le t \le \sum_{j=1}^{n} p_j} F_n(k, l, t)$ and the optimal schedule is obtained by backtracking. The time complexity of this algorithm is $O(n^3 B^2 \sum_{j=1}^{n} p_j)$.

5. Conclusions

In this paper we investigated the problem of scheduling jobs with agreeable processing times and due dates on a single batch processing machine. Lawler [8] showed that the problem $1|\mathrm{agr}(p_j, d_j)|\sum w_j T_j$ is strongly NP-hard. Therefore, the problem $1|B, \mathrm{agr}(p_j, d_j)|\sum w_j T_j$ is strongly NP-hard, too. We proved that the problem $1|B, \mathrm{agr}(p_j, d_j)|\sum T_j$ is NP-hard even if $B = 2$ and $p_{i+1} - p_i \le d_{i+1} - d_i$ ($i = 1, \dots, n-1$), and provided pseudo-polynomial-time algorithms for the problems $1|B, \mathrm{agr}(p_j, d_j)|\sum T_j$ with $B = 2$ and $p_{i+1} - p_i \le d_{i+1} - d_i$ ($i = 1, \dots, n-1$) and $1|B, \mathrm{agr}(p_j, d_j)|\sum w_j U_j$, respectively.

To find efficient algorithms for the problem $1|B = 2, \mathrm{agr}(p_j, d_j)|\sum T_j$ and the even more general problem $1|B, \mathrm{agr}(p_j, d_j)|\sum T_j$ are very challenging topics for future research.

Acknowledgements

The authors are thankful to the referees for their helpful suggestions and comments on an earlier version of the paper. This research was supported in part by The Hong Kong Polytechnic University under a doctoral research studentship for Liu.

References

[1] P. Brucker, A. Gladky, H. Hoogevreen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn, S. Van de Velde, Scheduling a batching machine, Journal of Scheduling 1 (1998) 31–54.
[2] T.C.E. Cheng, M.C. Gupta, Survey of scheduling research involving due date determination decisions, European Journal of Operational Research 38 (1989) 156–166.
[3] T.C.E. Cheng, Z.H. Liu, W.C. Yu, Scheduling jobs with release dates and deadlines on a batch processing machine, IIE Transactions 33 (2001) 685–690.
[4] J.Z. Du, J.Y.-T. Leung, Minimizing total tardiness on one machine is NP-hard, Mathematics of Operations Research 15 (1990) 483–495.
[5] H. Emmons, One-machine sequencing to minimize certain functions of job tardiness, Operation Research 17 (1969) 701–715.
[6] Y. Ikura, M. Gimple, Efficient scheduling algorithms for a single batch processing machine, Operations Research Letters 5 (1986) 61–65.
[7] R.M. Karp, Reducibility among combnatorial problems, in: R.D. Miller, J.W. Thatcher (Eds.), Complexity in Computer Computations, Plenum Press, New York, 1972, pp. 85–103.
[8] E.L. Lawler, A “pseudopolynomial” algorithm for sequencing jobs to minimize total tardiness, Annals of Discrete Mathematics 1 (1977) 331–342.

[9] E.L. Lawler, J.M. Moore, A functional equation and its application to resource allocation and sequencing problems, Management Science 16 (1969) 77–84.
[10] C.Y. Lee, R. Uzsoy, L.A. Martin Vega, Efficient algorithms for scheduling semiconductor burn-in operations, Operation Research 40 (1992) 764–775.
[11] C.L. Li, C.Y. Lee, Scheduling with agreeable release times and due dates on a batch processing machine, European Journal of Operational Research 96 (1997) 564–569.
[12] Z.H. Liu, W.C. Yu, Scheduling one batch processor subject to job release dates, Discrete Applied Mathematics 105 (2000) 129–136.
[13] Z.H. Liu, J.J. Yuan, T.C.E. Cheng, On scheduling an unbounded batch machine, Operations Research Letters 31 (2003) 42–48.