Scheduling Start Time Dependent Tasks with Deadlines and Identical Initial Processing Times on a Single Machine

T. C. E. Cheng **Q. Ding**

Department of Management
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
E-mail: mscheng@polyu.edu.hk

July 2001

Abstract

In this paper we study the feasibility problem of scheduling a set of start time dependent tasks on a single machine with deadlines, processing rates and identical initial processing times. First, we show that the cases with arbitrary deadlines are strongly NP-complete. Second, we show that the cases with two distinct deadlines are NP-complete in the ordinary sense. Finally, we give an optimal polynomial algorithm for the makespan problem with two distinct processing rates. We solve a series of open problems in the literature and give a sharp boundary delineating the complexity of the problems.

Keywords: Sequencing, Time dependence scheduling, Computational complexity.

1. Introduction

Machine scheduling problems with start time dependent processing times have received increasing attention in recent years. The linear model is one of the most popular ones. Formally, these problems can be stated as follows. A task system consists of *n* independent tasks and is denoted by $TS = (\{T_i\}, \{d_i\}, \{a_i\}, \{b_i\})$. Each task T_i is associated with a deadline d_i and characterized by an initial processing time $a_i \geq 0$ and a processing rate $b_i \geq 0$. Depending on the task starting time s_i, the actual processing time of task T_i is $p_i = a_i \pm b_i s_i \geq 0$ and its completion time is $C_i = s_i + p_i = a_i + (1 \pm b_i)s_i$. For all tasks, the release time is 0. Since the processing times are not integers in many practical cases, values a_i, b_i and d_i are allowed to be rational numbers. Adopting the three-field notation proposed by Graham et al [9] to describe a scheduling problem, we denote the makespan problem as $1/p_i = a_i \pm b_i s_i, d_i / C_{\max}$.

A nonpreemptive schedule is *feasible* if each T_i is completely processed in the interval $[0, d_i]$. The *feasibility problem* is to decide whether there exists a feasible schedule for *TS*. Let $C_{\max} = \max_{1 \leq i \leq n} \{C_i\}$ denote the maximum completion time of a schedule and $G = \max_{1 \leq i \leq n} \{d_i\} \equiv d_{\max}$ denote a threshold. The feasibility problem is equivalent to the decision version of the makespan problem, which is to decide whether there exists a feasible schedule for *TS* with $C_{\max} \leq G$.

Several papers consider the makespan problems of single machine scheduling with start time dependent tasks. For the model $p_i = a_i + b_i s_i \geq 0$ without deadlines, Gupta and Gupta [10] show that the schedule of tasks arranged in nondecreasing order of the ratios $\frac{a_i}{b_i}$ is optimal for the makespan problem. Meanwhile, Browne and Yechiali [2] obtain a similar result for a stochastic version of the problem. Cheng and Ding [7] show that the makespan problem with arbitrary deadlines is strongly NP-complete, and the case with $b_i = b$ can be solved in $O(n^5)$ time. Kononov [12] shows that the maximum lateness problem is NP-complete in the ordinary sense even with $a_i = 0$ for all tasks except one, and the due dates $d_i = d$ for all tasks with $a_i = 0$.

For the model $p_i = a_i - b_i s_i \geq 0$, Ho et al [11] show that the schedule of tasks arranged in nonincreasing order of the ratios $\frac{a_i}{b_i}$ is optimal for the makespan problem with identical deadlines.

Cheng and Ding show that the case with $b_i = b$ and arbitrary deadlines is strongly NP-complete [6], while the case with $b_i = b$ and two distinct deadlines is NP-complete in the ordinary sense [4]. Woeginger [14] and Chen [3] give two dynamic programming algorithms to solve the number of late task problem with identical deadlines in $O(n^3)$ and $O(n^2)$ time, respectively. The more recent results for scheduling models with time dependent processing times are surveyed in Alidaee and Womer [1].

The scheduling problem with identical processing times is an important branch in classical scheduling theory (see the results surveyed in Tanaev et al [13]). However, the corresponding problem with time dependent processing times is a virtually new area of study. In this paper, we focus on the complexity of the feasibility problems with arbitrary processing rates and identical initial processing times, i.e., $a_i = a$. We show that the cases with arbitrary processing rates and arbitrary deadlines, denoted as $1/p_i = a \pm b_i s_i, d_i / C_{\max}$, are strongly NP-complete. We also show that the cases with arbitrary processing rates and two distinct deadlines, denoted as $1/p_i = a \pm b_i s_i, d_i \in \{D_1, D_2\} / C_{\max}$, are NP-complete in the ordinary sense. Finally, we give an optimal polynomial algorithm for the makespan problem with only two distinct processing rates, denoted as $1/p_i = a \pm b_i s_i, d_i, b_i \in \{B_1, B_2\} / C_{\max}$.

The considered models are rich in practical applications. Ho et al [11] introduce an interesting military application with negative processing rates. The task is to destroy an aerial threat and its execution time decreases with time as the longer the action is delayed, the closer the threat gets. This application is actually a problem with $a_i = a$, considering that the efficient scope of the action is usually a sphere. For the case with positive processing rates and $a_i = a$, we consider medical treatment as an application example. At the outset, the treatment is common to the patients that take an identical processing time. However, if the treatment is delayed, additional efforts are needed for each treated individual, resulting in a longer time for each subsequent treatment.

For the model $p_i = a_i - b_i s_i \geq 0$, not only p_i, but also C_i is decreasing in s_i if $b_i > 1$. In such a case, some performance measures become non-regular. We assume $0 \leq b_i \leq 1$ in this paper. Ho et al [11] make some additional assumptions, such as $b_i d_i < a_i \leq d_i$, which are reasonable and indeed help eliminate some uninteresting cases. However, they are not necessary for the results in this paper. On the other hand, a similar model with arbitrary processing rates is considered in Cheng et al [5].

Furthermore, only schedules without idle time need to be considered. Without affecting the results of NP-completeness, in Sections 2 and 3, we assume that the identical initial processing time is 1, i.e., $a = 1$, is used in the formula describing the completion time of a task.

2. NP-completeness of the problems with arbitrary deadlines

The strongly NP-complete 3-Partition problem (see Garey and Johnson [8]) can be reduced to $1/p_i = 1 - b_i s_i, d_i / C_{\max}$.

3-Partition. Given a list $H = \{h_1, h_2, \cdots, h_{3m}\}$ of $3m$ integers such that $\sum_{i=1}^{3m} h_i = mB$ and $\frac{B}{4} < h_i < \frac{B}{2}$ for each $1 \le i \le 3m$, can H be partitioned disjointedly into $H_1, H_2, \cdots, H_m$ such that $\sum_{h_i \in H_j} h_i = B$ for each $1 \le j \le m$?

Given an instance I of 3-Partition with a list $H = \{h_1, h_2, \cdots, h_{3m}\}$ and B, define $q = 2mB$, $v = 32m^2 qB$. Construct an instance II of $1/p_i = 1 - b_i s_i, d_i / C_{\max}$ as follows.

The set of tasks is $TS = V \cup R \cup Q_1 \cup \cdots \cup Q_{m-1}$, where $V = \{T_{0,1}, T_{0,2}, \cdots, T_{0,v}\}$, $R = \{T_1, T_2, \cdots, T_{3m}\}$, $Q_i = \{T_{i,1}, T_{i,2}, \cdots, T_{i,q}\}$, for $1 \le i \le m-1$. Define $A_1 = 3n^3$ and $A_2 = A_3 = 2nmB$, where $n = v + 3m + (m-1)q$ is the number of tasks in TS .

Given the threshold $G = n - \sum_{k=1}^{m-1} \sum_{l=1}^{q} \frac{v + qk + 3k - l}{A_1 A_3} - \sum_{k=0}^{m-1} \frac{(v + qk + 3k)B}{A_1 A_2 A_3}$ and the constants $D_i = v + qi + 3i - \sum_{k=1}^{i-1} \sum_{l=1}^{q} \frac{v + qk + 3k - l}{A_1 A_3} - \sum_{k=0}^{i-1} \frac{(v + qk + 3k)B}{A_1 A_2 A_3}$, for $1 \le i \le m-1$. We assume that the processing rates and the deadlines are

$b_{0,i} = 0,\ d_{0,i} = v,$ for $1 \le i \le v$,

$b_{i,j} = \frac{1}{A_1 A_3},\ d_{i,j} = D_i$, for $1 \le i \le m-1$ and $1 \le j \le q$,

$b_i = \dfrac{h_i}{A_1 A_2 A_3}$, $d_i = G$, for $1 \le i \le 3m$.

Now, we analyze the structure of the feasible schedule for II (see Figure 1). Let S be a given feasible schedule for II. Since $d_{0,i} = v$, for $1 \le i \le v$, the tasks in V should be scheduled in the first v positions in S. Since the tasks in $Q_1 \cup \cdots \cup Q_{m-1}$ have identical processing rates, we can arrange them in nondecreasing order of their deadlines and indices. The set of tasks after $T_{0,v}$ and before $T_{1,q}$ consists of the tasks in Q_1 and some tasks in R. All of their deadlines are larger than D_1. We re-arrange the task set in nonincreasing order of the processing rates. Since the processing rates of tasks in R are smaller than the processing rates of tasks in Q_1, the resulting schedule is in the form of (R_1, Q_1). Since the schedule of tasks in nonincreasing order of the processing rates minimizes the total processing time (see Ho et al [11]), the resulting schedule is also feasible. Similarly, we can swap the tasks after $T_{i,q}$ and before $T_{i+1,q}$ in the form of (R_{i+1}, Q_{i+1}) for $1 \le i \le m-2$. Finally, we obtain a new feasible schedule in the form of $(V, R_1, Q_1, R_2, \cdots, Q_{m-1}, R_m)$, which is called a *basic* schedule. Thus, we obtain the following lemma.

Lemma 1. If there exists a feasible schedule for II, then there exists a feasible basic schedule. ■

In a basic schedule for II, if there are exactly three tasks in each R_j and $\sum_{T_i \in R_j} \boldsymbol{h_i} = \boldsymbol{B}$, for $1 \le j \le m$, then we obtain an ideal schedule, which is called a *standard* schedule. Now we illustrate that every standard schedule is feasible as follows.

Given a standard schedule S, since the processing rates are very small, all actual processing times are almost equal to $\mathbf{1}$. For any task $T_i \in R_j$ in S, the actual processing time is $p_i = 1 - b_i s_i = 1 - b_i \cdot [v + (j-1)q] - b_i \cdot \Delta s_i$, where $\Delta s_i = s_i - v - (j-1)q$ is much smaller than q. Since there are exactly three tasks in R_j and $\sum_{T_i \in R_j} \boldsymbol{h_i} = \boldsymbol{B}$, we have

$$\sum_{T_i \in R_j} p_i = 3 - \sum_{T_i \in R_j} b_i s_i = 3 - \frac{[v + (j-1)q]B}{A_1 A_2 A_3} - \sum_{T_i \in R_j} b_i \Delta s_i \,. \qquad (1)$$

Define $P_{R_j} = 3 - \frac{[v + (j-1)q]B}{A_1 A_2 A_3}$, for $1 \le j \le m$. Since the error term $\sum_{T_i \in R_j} b_i \Delta s_i$ is much smaller than $\frac{q}{A_1 A_2 A_3}$, from (1), we have

$$P_{R_j} - \frac{q}{4mA_1 A_2 A_3} \le \sum_{T_i \in R_j} p_i \le P_{R_j}. \quad (2)$$

If Q_j is scheduled consecutively and completed exactly at D_j, then the total actual processing time of Q_j is its minimum, denoted as P_{Q_j}, for $1 \le j \le m-1$.

Note that $D_1 = v + P_{R_1} + P_{Q_1}$; that is, if Q_1 is scheduled consecutively and completed exactly at D_1, then the length of the interval left for R_1 is exactly P_{R_1}. Such a sub-schedule (R_1, Q_1) is feasible. By induction, we can generalize the above results to the reminder of the schedule and obtain the following lemma.

Lemma 2. If a schedule for II is standard, then it is feasible. ■

On the other hand, if a basic schedule is not standard, then we can show that it is not feasible either. Given a basic schedule S, if there are more than $3j$ tasks in $R_1 \cup \cdots \cup R_j$, then, from (1), it is easy to see that $\sum_{T_i \in R_1 \cup \cdots \cup R_j} p_i > 3j + \frac{1}{2} > \sum_{i=1}^{j} P_{R_i}$. From the definition of D_j, the task set Q_j cannot meet D_j. Similarly, if there are exactly $3j$ tasks in $R_1 \cup \cdots \cup R_j$ and $\sum_{T_i \in R_1 \cup \cdots \cup R_j} h_i < jB$, then we also have that $\sum_{T_i \in R_1 \cup \cdots \cup R_j} p_i > \sum_{i=1}^{j} P_{R_i}$ and Q_j cannot meet D_j. Thus, we obtain the following remark.

Remark 1. For a feasible basic schedule, there are no more than $3j$ tasks in $R_1 \cup \cdots \cup R_j$ and $\sum_{T_i \in R_1 \cup \cdots \cup R_j} h_i \ge jB$, for $1 \le j \le m$. ■

Now, we analyze the converse cases. Given a basic schedule, suppose that there are only two tasks in R_1. Comparing with a standard schedule, the starting times of tasks in Q_1 are all earlier by

almost 1. The total actual processing time of tasks in Q_1 is near $P_{Q_1} + \frac{q}{A_1 A_2}$. Meanwhile, the total actual processing time of tasks in Q_j is always larger than its minimum P_{Q_j}. As to the tasks in R, similar to (2), we have

$$\sum_{T_i \in R} p_i - \sum_{i=1}^{m} P_{R_i} \le \sum_{T_i \in R} b_i \cdot (s_i - v) \le \frac{3m \cdot B \cdot 2mq}{A_1 A_2 A_3}. \tag{3}$$

Since $\frac{q}{A_1 A_2}$ is much larger than $\frac{q(3m \cdot B \cdot 2m + 1)}{A_1 A_2 A_3}$, from (3) and the definition of G, we get $C_{\max} > G$; that is, the schedule is not feasible. Thus, we have that there are exactly three tasks in R_1. By induction and Remark 1, we obtain the following remark.

Remark 2. For a feasible basic schedule, there are exactly $3j$ tasks in $R_1 \cup \cdots \cup R_j$ and $\sum_{T_i \in R_1 \cup \cdots \cup R_j} h_i \ge jB$, for $1 \le j \le m$. ■

Given a feasible basic schedule, from Remark 2, there are exactly $3j$ tasks in $R_1 \cup \cdots \cup R_j$ and $\sum_{T_i \in R_1 \cup \cdots \cup R_j} h_i \ge jB$, for $1 \le j \le m$. Suppose that

$$\sum_{T_i \in R_1} h_i = B + 1, \quad \sum_{T_i \in R_2 \cup \cdots \cup R_{j-1}} h_i = B \text{ and } \sum_{T_i \in R_j} h_i = B - 1. \tag{4}$$

Similar to (1), we have

$$\sum_{T_i \in R_1} p_i \approx 3 - \frac{v(B+1)}{A_1 A_2 A_3} \tag{5}$$

and

$$\sum_{T_i \in R_j} p_i \approx 3 - \frac{[v + (j-1)q](B-1)}{A_1 A_2 A_3}. \tag{6}$$

From (5) and (6), we have

$$\sum_{T_i \in R_1 \cup R_j} p_i - (PR_1 + PR_j) \approx \frac{(j-1)q}{A_1 A_2 A_3} \ge \frac{q}{2A_1 A_2 A_3}. \tag{7}$$

If the other parts of the given schedule are in the standard form, then, from (2), we have $\sum_{T \notin R_1 \cup R_j, T_i \in R} p_i - \sum_{i \ne j, 2 \le j \le m} P_{R_i} \ge -\frac{q}{4A_1 A_2 A_3}$. Further, from (7) and the definition of G, we have that

$\sum_{T_i \in R} p_i - \sum_{1 \le j \le m} P_{R_i} \ge \frac{q}{4A_1A_2A_3}$ and $C_{\max} > G$, contradicting the feasibility assumption. Thus, the given schedule is standard. If there exists another sub-schedule which is not in the standard form, then, from Remark 2, the structure of this sub-schedule should fulfil the assumption for the originally given schedule. Repeating the above analysis and results, we obtain the following lemma.

Lemma 3. If a basic schedule for II is feasible, then it is standard. ■

Given a solution for I, we can construct a corresponding standard schedule. From Lemma 2, it is feasible. Given a feasible schedule for II, from Lemmas 1 and 3, there exists a feasible basic schedule, which is standard and corresponds to a solution for I. A similar reduction with a detailed proof is presented in Cheng and Ding [6]. Thus, we obtain the following theorem.

Theorem 1. The problem of whether there exists a feasible schedule for the problem $1/p_i = 1 - b_i s_i, d_i / C_{\max}$ is strongly NP-complete. ■

A similar technique can be used to tackle $1/p_i = 1 + b_i s_i, d_i / C_{\max}$. Given an instance I of 3-Partition with a list $H = \{h_1, h_2, \cdots, h_{3m}\}$ and B, define $q = 32m^2B$, $v = 16m^2qB$ and construct an instance II of $1/p_i = 1 + b_i s_i, d_i / C_{\max}$ as follows.

The set of tasks is $TS = V \cup R \cup Q_1 \cup \cdots \cup Q_{m-1}$, where $V = \{T_{0,1}, T_{0,2}, \cdots, T_{0,v}\}$, $R = \{T_1, T_2, \cdots, T_{3m}\}$ and $Q_i = \{T_{i,1}, T_{i,2}, \cdots, T_{i,q}\}$, for $1 \le i \le m-1$. Define $E = 4mnB$ and $A = 32n^3E^2$, where $n = v + 3m + (m-1)q$ is the number of tasks in TS.

Given the threshold $G = n + \sum_{k=0}^{m-1} \frac{3E(v + qk + 3k + 1)}{A} + \sum_{k=0}^{m-1} \frac{B(v + qk + 3k + 1)}{A} + \frac{2mB}{A}$ and the constants $D_i = v + qi + 3i + \sum_{k=0}^{i-1} \frac{3E(v + qk + 3k + 1)}{A} + \sum_{k=0}^{i-1} \frac{B(v + qk + 3k + 1)}{A} + \frac{2mB}{A}$, for $1 \le i \le m-1$. We assume that the processing rates are

$b_{0,i} = 0,\ d_{0,i} = v$, for $1 \le i \le v$,

$b_{i,j} = 0,\ d_{i,j} = D_i$, for $1 \le i \le m-1$ and $1 \le j \le q$,

$b_i = \dfrac{E + h_i}{A}$, $d_i = G$, for $1 \le i \le 3m$.

Let S be a given feasible schedule for II. Since $d_{0,i} = v$, for $1 \le i \le v$, the tasks in V should be scheduled in the first v positions. Since the schedule of tasks arranged in nonincreasing increasing processing rates minimizes the total processing time (see Gupta and Gupta [10]) and the increasing processing rates of tasks in R are larger than those of tasks in $Q_1 \cup \cdots \cup Q_{m-1}$, the tasks in R should be scheduled as early as possible. By swapping tasks in $Q_1 \cup \cdots \cup Q_{m-1}$ and R, we obtain a feasible schedule in the form of $(V, R_1, Q_1, R_2, \cdots, Q_{m-1}, R_m)$. Moreover, the number of tasks in each R_j is exactly three and the sum of the processing rates is exactly $\dfrac{3E + B}{A}$. Similar to Theorem 1, we obtain the following theorem.

Theorem 2. The problem of whether there exists a feasible schedule for the problem $1/p_i = 1 + b_i s_i, d_i / C_{\max}$ is strongly NP-complete. ■

3. NP-completeness of the problems with two distinct deadlines

Now we discuss the complexity issue for cases with two distinct deadlines. The NP-complete Partition problem (Garey and Johnson [8]) can be reduced to $1/p_i = 1 - b_i s_i, d_i \in \{D_1, D_2\} / C_{\max}$.

Partition. Given a list $H = \{h_1, h_2, \cdots, h_m\}$ of m integers such that $\sum_{i=1}^{m} h_i = 2B$, can H be partitioned disjointedly into H_1 and H_2 such that $\sum_{h_i \in H_1} h_i = \sum_{h_i \in H_2} h_i = B$?

Given an instance I of Partition with a list $H = \{h_1, h_2, \cdots, h_m\}$ and B, construct an instance II of $1/p_i = 1 - b_i s_i, d_i \in \{D_1, D_2\} / C_{\max}$ as follows.

The set of tasks is $TS = R_0 \cup R_1 \cup \cdots \cup R_m$, where $R_i = \{T_{i,0}, T_{i,1}, T_{i,2}, \cdots, T_{i,m+1}\}$, for $0 \le i \le m$. Define $A_1 = 4n^3$ and $A_2 = A_3 = 2^{m+1} m^m n^2 B$, where $n = (m+1)(m+2)$ is the number of tasks in TS. We assume that the processing rates are

$$b_{0,0} = b_{0,1} = 0,\ b_{0,j} = \frac{1}{A_1 A_3},\ \text{for } 2 \le j \le m+1,$$

$$b_{i,0} = \frac{2^i m^i B - h_i}{(i+1)A_1 A_2 A_3},\ b_{i,j} = \frac{2^i m^i B}{(i+1)A_1 A_2 A_3},\ \text{for } 1 \le i \le m \text{ and } 1 \le j \le m+1.$$

Define

$$D_1 = 2m + 2 - \sum_{i=1}^{m} (i+1) b_{i,1} - \sum_{j=2}^{m+1} (m+j) b_{0,j} + \frac{B}{A_1 A_2 A_3} + \frac{1}{2A_1 A_2 A_3}$$

and

$$D_2 = n - \sum_{i=1}^{m} (i+1) b_{i,1} - \sum_{j=2}^{m+1} (m+j) b_{0,j} - \sum_{i=1}^{m} (i+1)(m+1) b_{i,0}$$
$$- \sum_{i=1}^{m} \sum_{j=1}^{m} [(i+1)(m+1) + j] b_{i,j+1} - \frac{mB}{A_1 A_2 A_3} + \frac{1}{2A_1 A_2 A_3}.$$

The deadlines are $d_{0,j} = D_1$ and $d_{i,j} = D_2$, for $1 \le i \le m$ and $0 \le j \le m+1$. The threshold is $G = D_2$.

If there exists a feasible schedule, then, using a strategy similar to the above two reductions, we get a basic feasible schedule in the form of $R_{0,1}(R_{1,1}, R_{2,1}, \cdots, R_{m,1}) R_{0,2}(R_{1,2}, R_{2,2}, \cdots, R_{m,2})$, where $R_{i,1} \cup R_{i,2} = R_i$, for $0 \le i \le m$. Moreover, if a basic schedule is feasible, then we have $R_{0,1} = \{T_{0,0}, T_{0,1}\}$ and there is exactly one task, either $T_{i,0}$ or $T_{i,1}$, in $R_{i,1}$. That is, the schedule is in a standard form such as $(T_{0,0}, T_{0,1})(T_{1,k_1}, T_{2,k_2}, \cdots, T_{m,k_m})(R_{0,2})(R_{1,2}, R_{2,2}, \cdots, R_{m,2})$, $k_i = 0$ or 1, (see Figure 2). The total actual processing time of a schedule with $k_i = 0$ is different from that of the counterpart case with $k_i = 1$ by a multiple of h_i. The multiple for each index i is equal, ensured by the structure of II. For the case $\sum_{i \in \{j : k_j = 0\}} h_i > B$, the sum of processing rates before $R_{0,2}$ is so small and the total actual processing time before $R_{0,2}$ is so large that $R_{0,2}$ cannot meet D_1. On the other hand, for the case $\sum_{i \in \{j : k_j = 0\}} h_i < B$, the sum of processing rates after $R_{0,2}$ is so small and the total actual

processing time after $R_{0,2}$ is so large that we have $C_{\max} > G = D_2$. Each standard feasible schedule for *II* corresponds to a solution for *I*. A similar reduction with a detailed proof is presented in Cheng and Ding [4]. Thus, similar to Theorems 1 and 2, we obtain the following theorem.

Theorem 3. The problem of whether there exists a feasible schedule for the problem $1/p_i = 1 - b_i s_i, d_i \in \{D_1, D_2\}/C_{\max}$ is NP-complete in the ordinary sense. ■

An instance *II* of $1/p_i = 1 + b_i s_i, d_i \in \{D_1, D_2\}/C_{\max}$ is constructed as below: *TS* consists of $n = (m+2)(m+1)$ tasks: $TS = \{T_{0,0}, T_{0,1}, \cdots, T_{0,m+1}\} \cup \{T_{1,0}, T_{1,1}, \cdots, T_{1,m+1}\} \cup \cdots \cup \{T_{m,0}, T_{m,1}, \cdots, T_{m,m+1}\}$. Define $E = n^2 2^{2m} m^{2m} B$ and $A = 16n^3 E^2$. The processing rates are

$$b_{0,0} = b_{0,1} = \frac{2E}{A},\ b_{0,j} = 0, \text{ for } 2 \le j \le m+1,$$

$$b_{i,0} = \frac{E + 2^{2m-2i+2} m^{2m-2i+2} B + h_i}{(i+1)A},\ b_{i,j} = b_{i,0} - \frac{h_i}{(i+1)A}, \text{ for } 1 \le i \le m,\ 1 \le j \le m+1.$$

Define

$$D_1 = 2m + 2 + \frac{4E - 2B + 1}{2A} + \sum_{i=1}^{m} (i+1) b_{i,0}$$

and

$$D_2 = n + \frac{4E + 2mB + 1}{2A} + \sum_{i=1}^{m} (i+1) b_{i,0} + \sum_{i=1}^{m} \sum_{j=0}^{m} [(i+1)(m+1) + j] b_{i,j+1}.$$

The deadlines are $d_{0,j} = D_1$ and $d_{i,j} = D_2$, for $1 \le i \le m$ and $0 \le j \le m+1$. The threshold is $G = D_2$. Similar to Theorem 3, we obtain the following theorem.

Theorem 4. The problem of whether there exists a feasible schedule for the problem $1/p_i = 1 + b_i s_i, d_i \in \{D_1, D_2\}/C_{\max}$ is NP-complete in the ordinary sense. ■

4. Solvable cases

Now we present a polynomial optimal algorithm for the makespan problem $1/p_i = a \pm b_i s_i, d_i, b_i \in \{B_1, B_2\}/C_{\max}$. In this algorithm, we try to generate a feasible schedule, in

which the tasks with the smaller processing rate B_1 are scheduled as early as possible and the tasks with the larger processing rate B_2 are scheduled as late as possible (see Figure 3).

Given an instance I of $1/p_i = a - b_i s_i, d_i, b_i \in \{B_1, B_2\}/C_{\max}$ with m distinct deadlines $D_1 < D_2 < \cdots < D_m$. A schedule is called *canonical*, if it satisfies that the tasks with the same b_i are in the early due date (EDD) order and the tasks in $R_i = \{T_i : C_i \in (D_{i-1}, D_i]\}$, where $1 \le i \le m$ and $D_0 = 0$, are in nondecreasing b_i order. If there exists a feasible schedule for a given instance, then, similar to Lemma 1, we can get the following lemma.

Lemma 4. If there exists a feasible (optimal) schedule for an instance of $1/p_i = a - b_i s_i, d_i, b_i \in \{B_1, B_2\}/C_{\max}$, then there exists a canonical feasible (optimal) schedule. ■

Now, we introduce the algorithm. Schedule the tasks with the same b_i in EDD order and we get two task chains S_1 and S_2. Insert the tasks in S_2 into $[0, D_m]$ from backward as follows. Schedule the tasks in S_2 with the largest deadline D_m as a consecutive subschedule $S_{2,m}$ completed at D_m. Schedule the tasks in S_2 with D_{m-1} as a consecutive subschedule $S_{2,m-1}$ completed at the minimum of D_{m-1} and the stating time of $S_{2,m}$. By induction, we can insert the remainder tasks in S_2 and generate a schedule in the form of $(I_1, M_1)(I_2, M_2)\cdots(I_k, M_k)$, where I_i is the i-th idle time and M_i is the i-th consecutive subschedule, $k \le m$. This schedule is called the *middle schedule*, denoted as M. If some tasks in S_2 cannot be inserted in M, then put them in *a late task set*, denoted as L. Then, we insert the tasks in S_1 into the idle times in M from forward. Starting from the beginning of I_1, insert the tasks according to the order in S_1, until the spare space of I_1 cannot hold the next task. If an inserted task cannot meet its deadline, then take it out of the schedule and put it in L. Then, insert the following task continuously. Let $S_{1,1}$ denote the inserted consecutive subschedule. Shift M_1 forward to connect with $S_{1,1}$. Apply the same operations to the remaining schedule. We get a *final schedule* F in the form of $(S_{1,1}, M_1)(S_{1,2}, M_2)\cdots(S_{1,k}, M_k)$ and a late task set L. The following two lemmas are used to show that the above algorithm is optimal.

Lemma 5. For an instance of $1/p_i = a - b_i s_i, d_i, b_i \in \{B_1, B_2\}/C_{\max}$, the final schedule is optimal if the late task set is empty.

Proof. Since L is empty, all tasks with $b_i = B_2$ are scheduled in M. From the generation of F, F is a feasible schedule for II and the tasks in M and S_1 are inserted in EDD order, respectively. From Lemma 4, there exists a canonical optimal schedule for II, denoted as S. In S, the tasks with the same b_i are also in EDD order. In S before the last task in M_1, the tasks with $b_i = B_1$ are no more than the tasks in $S_{1,1}$. Otherwise, from the generation of $S_{1,1}$, the last task in M_1 cannot meet its deadline in S, a contradiction.

In S, advancing the tasks in $S_{1,1}$ to the beginning of schedule, the resulting schedule S^* starts with $(S_{1,1}, M_1)$. Since the tasks with the smaller processing rate $b_i = B_1$ always shift forward, the makespan of S^* is no larger than the makespan of S. A task with $b_i = B_2$ before a task in $S_{1,1}$ may shift backward, but its completion time is still no larger than that in F. Hence, S^* is also an optimal schedule. Applying the above policy to the remaining part of S, we finally get an optimal schedule with the same task order as F. That is, the final schedule F is optimal. ■

Lemma 6. If the late task set is not empty, then an instance of $1/p_i = a - b_i s_i, d_i, b_i \in \{B_1, B_2\}/C_{\max}$ does not have a feasible schedule.

Proof. Let T^* denote the first late task generated by the algorithm. For the case with $b^* = B_2$, there exists a continuous sub-schedule consisting of $(M_1, \cdots, M_j)$, which cannot be inserted in $\lfloor 0, D_j \rfloor$, according to the generation of M and L. The makespan of this sub-instance is independent of the order of tasks, since they have identical initial processing times and processing rates. Thus, the instance has no feasible schedule.

For the case with $b^* = B_1$, according to the algorithm, the tasks with $b_i = B_2$ and the tasks before T^* in S_1 have been inserted in a partial schedule F^*. However, T^* cannot be inserted in F^*. On the other hand, if the instance has a feasible schedule, then it has a canonical feasible schedule F, in which the task order is the same as that of the tasks in F^*, according to the proof in Lemma 5. This means that T^* can be inserted in F^*, a contradiction. ■

From Lemmas 5 and 6, we obtain the following optimal algorithm for $1/p_i = a - b_i s_i, d_i, b_i \in \{B_1, B_2\}/C_{\max}$.

Algorithm A: for $1/p_i = a - b_i s_i, d_i, b_i \in \{B_1, B_2\}/C_{\max}$

1. Establish task chains S_1 and S_2 .
2. Establish the middle schedule M , the final schedule F and the late task set L .
3. If L is empty, then F is optimal. Otherwise, there exists no feasible schedule.

Since each step in Algorithm A can be completed in $O(n \log n)$ time, the total running time of Algorithm A is $O(n \log n)$. This method can be readily adapted to deal with $1/p_i = a + b_i s_i, d_i, b_i \in \{B_1, B_2\}/C_{\max}$. Thus, we obtain the following theorem.

Theorem 5. The makespan problem $1/p_i = a \pm b_i s_i, d_i, b_i \in \{B_1, B_2\}/C_{\max}$ can be solved in $O(n \log n)$ time. ■

5. Conclusions

In this paper, we have examined the single machine scheduling problems with start time dependent tasks that have identical initial processing times. We have shown that the feasibility problems, $1/p_i = 1 \pm b_i s_i, d_i/C_{\max}$, are strongly NP-complete, and the feasibility problems with two distinct deadlines, $1/p_i = 1 \pm b_i s_i, d_i \in \{D_1, D_2\}/C_{\max}$, are NP-complete in the ordinary sense. As usually the case, all these results can be adapted to their corresponding scheduling problems with the popular regular performance measures of total flow time and maximum lateness.

On the other hand, we have shown that the makespan problem $1/p_i = a \pm b_i s_i, d_i, b_i \in \{B_1, B_2\}/C_{\max}$ can be solved in $O(n \log n)$ time. For the general makespan problem $1/p_i = a \pm b_i s_i, d_i/C_{\max}$, it is not difficulty to develop a heuristic based on the intuition of Algorithm A or design an exact algorithm by enumerating all canonical schedules. The combination

of these approaches should be effective on average, but it is hard to get an ideal upper bound for the error or running time. For further research, it would be interesting to design approximation algorithms with a constant worst-case bound for the problems or prove infeasibility of such approximations.

Acknowledgements

We are grateful to two anonymous referees for their constructive comments on an earlier version of this paper. This research is supported in part by The Hong Kong Polytechnic University under grant number 350/239. The first author is partially supported by the Croucher Foundation under a Croucher Senior Research Fellowship.

References

[1]. B. Alidaee and N. K. Womer (1999). Scheduling with time dependent processing times: review and extensions, *Journal of the Operational Research Society*, **50**, 711-720.

[2]. S. Browne and U. Yechiali (1990). Scheduling deteriorating jobs on a single processor, *Operations Research*, **38**, 495-498.

[3]. Z.-L. Chen (1995). A note on single-processor scheduling with time-dependent execution times, *Operations Research Letters*, **17**, 127-129.

[4]. T. C. E. Cheng and Q. Ding (1998). The complexity of single machine scheduling with two distinct deadlines and identical decreasing rates of processing times, *Computers and Mathematics with Applications*, **35**, 95-100.

[5]. T. C. E. Cheng, Q. Ding and M. Y. Kovalyov (1998). Scheduling jobs with linearly decreasing processing times, *Working Paper*, Faculty of Business and Information Systems, The Hong Kong Polytechnic University, Hong Kong.

[6]. T. C. E. Cheng and Q. Ding (1999). The time dependent machine makespan problem is strongly NP-complete, *Computers and Operations Research*, **26**, 749-754.

[7]. T. C. E. Cheng and Q. Ding (2000). Single machine scheduling with deadlines and increasing rates of processing times, *Acta Informatica*, **36**, 673-692.

[8]. M. R. Garey and D. S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., New York.

[9]. R. L. Graham, E. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan (1976). Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, **5**, 287-326.

[10]. J. N. D. Gupta and S. K. Gupta (1988). Single facility scheduling with nonlinear processing times, *Computers and Industrial Engineering*, **14**, 387-393.

[11]. K. I-J. Ho, J. Y-T. Leung and W-D. Wei (1993). Complexity of scheduling tasks with time-dependent execution times, *Information Processing Letters*, **48**, 315-320.

[12]. A. V. Kononov (1997). Scheduling problems with linear increasing processing times, *Operations Research Proceedings 1996*: *Selected Papers of the Symposium on Operations Research (SOR '96)*, Braunschweig, Sept. 3-6, 1996, 208-212, Springer, Berlin.

[13]. V. S. Tanaev, V. S. Gordon and Y. M. Shafransky (1994). *Scheduling Theory, Single–stage Systems*, Kluwer, Dordrecht.

[14]. G. J. Woeginger (1995). Scheduling with time dependent execution times, *Information Processing Letters*, **54**, 155-156.

V	R_1	Q_1	...	R_{m-1}	Q_{m-1}	R_m
$T_{0,1},\cdots,T_{0,v}$	T_{i_1},T_{i_2},T_{i_3}	$T_{1,1},\cdots,T_{1,q}$	...	$T_{i_{3m-5}},T_{i_{3m-4}},T_{i_{3m-3}}$	$T_{m-1,1},\cdots,T_{m-1,q}$	$T_{i_{3m-2}},T_{i_{3m-1}},T_{i_{3m}}$

$d_{0,j}=v$ $\quad$ $d_{1,j}=D_1$ $\quad$ $d_{m-1,j}=D_{m-1}$ $\quad$ $d_i=G$

Figure 1. The basic and standard schedule of II for $1/p_i=1\pm b_i s_i, d_i/C_{\max}$

$R_{0,1}$	$R_{1,1}$	...	$R_{m,1}$	$R_{0,2}$	$R_{1,2}$	$\cdots$	$R_{m,2}$
$T_{0,0},T_{0,1}$	T_{1,k_1}	...	T_{m,k_m}	$T_{0,2},T_{0,2},\cdots,T_{0,m+1}$	$\{T_{1,0},T_{1,1},\cdots,T_{1,m+1}\}$ $-T_{1,k_1}$	$\cdots$	$\{T_{m,0},\cdots,T_{m,m+1}\}$ $-T_{1,k_m}$

$d_{0,j}=D_1$ $\quad$ $d_{i,j}=D_2=G$

Figure 2. The basic and standard schedule of II for $1/p_i=1\pm b_i s_i, d_i\in\{D_1,D_2\}/C_{\max}$

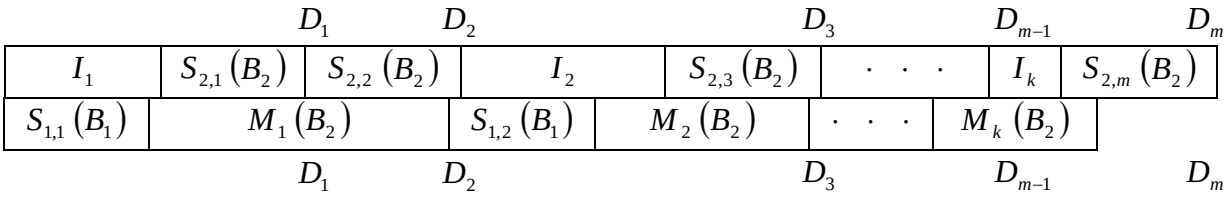

Figure 3. The middle and final schedule of Algorithm A