# Single-machine scheduling with trade-off between number of tardy jobs and compression cost *

Yong He[1, 2, †]

[1] Department of Mathematics, Zhejiang University, Hangzhou 310027, P.R. China
[2] State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, P.R. China

Qi Wei[3 ‡]

[3] Ningbo Institute of Technology, Zhejiang University, Ningbo 315100, P.R. China

T. C. E. Cheng [4, §]

[4] Department of Logistics, The Hong Kong Polytechnic University, Kowloon, Hong Kong

**Abstract**

We consider a single-machine scheduling problem in which the job processing times are controllable or compressible. The performance criteria are the compression cost and the number of tardy jobs. For the problem where no tardy jobs are allowed and the objective is to minimize the total compression cost, we present a strongly polynomial time algorithm. For the problem to construct the trade-off curve between the number of tardy jobs and the maximum compression cost, we present a polynomial time algorithm. Furthermore, we extend the problem to the case of discrete controllable processing times where the processing

*This research was supported by the Teaching and Research Award Program for Outstanding Young Teachers in Higher Education Institutions of the MOE, China, and the National Natural Science Foundation of China (10271110). The third author was supported in part by The Hong Kong Polytechnic University under a grant from the *ASD in China Business Services*.

†Email: mathhey@zju.edu.cn

‡Email: weiqi@nit.zju.edu.cn

§Corresponding author; email: LGTcheng@polyu.edu.hk

time of a job can only take one of some given discrete values. We show that even some special cases of the discrete controllable version with the objective of minimizing the total compression cost are NP-hard, but the general case is solvable in pseudo-polynomial time. Moreover, we present a strongly polynomial time algorithm to construct the trade-off curve between the number of tardy jobs and the maximum compression cost for the discrete controllable version.

# 1 Introduction

Most classical scheduling problems assume that the job processing times are fixed, i.e., they are determined a priori and cannot be modified by the scheduler. However, in real applications the processing of a job often requires resources, such as manpower, facilities, funds, raw materials, etc. By putting more resources into job processing, shorter job processing times may be accomplished. Hence, it is possible to compress jobs (control the job processing times) by incurring extra costs. Scheduling problems with controllable processing times have received much attention from researchers in the last two decades. Work in this area was initiated by Vickson [11, 12] and Van Wassenhove and Baker [13]. For the related work on machine scheduling problems with controllable processing times, the reader is referred to the survey by Nowicki and Zdrzalka [9]. In this paper we consider a single-machine model of joint job sequencing and resource allocation with the sequencing criterion being the number of tardy jobs, which was first proposed by Daniels and Sarin [5]. Formally, this problem can be formulated as follows.

We are given a set $J = \{J_1, J_2, \cdots, J_n\}$ of independent jobs, which must be scheduled on a single machine. The processing requirement of a job $J_i$ is specified by four non-negative parameters $p_i, d_i, u_i$ and $w_i$. Here $p_i$ denotes the normal (initial) processing time of $J_i$ , $d_i$ denotes the due-date of $J_i$, $u_i \leq p_i$ is an upper bound on the compressibility of $J_i$, and $w_i$ is the cost for unit-time compression of $J_i$, called the *unit-compression cost*. By allocating additional resources to the processing of $J_i$, the *actual* processing time of $J_i$ may be compressed from $p_i$ to $p'_i = p_i - x_i$, where $x_i \in [0, u_i]$ is the amount of compression. The cost of performing this compression equals $w_i x_i$. Let $C_i$ denote the completion time of job $J_i$ under a given schedule. Define $U_i = 0$ if job $J_i$ is early ($C_i < d_i$) or on-time ($C_i = d_i$), and $U_i = 1$ if it is tardy ($C_i > d_i$). We assume that all the jobs are available at time zero. Moreover, let $TCC = \Sigma_{i=1}^{n} w_i x_i$ be the total compression cost, and $n_T = \Sigma_{i=1}^{n} U_i$ the number of tardy jobs. Then the objective is to construct the trade-off curve between $n_T$ and $TCC$. We denote this problem by $P_0 : 1 \mid contr, n_T \leq k \mid TCC$ in this paper.

For this problem, Daniels and Sarin [5] provided some theoretical results. Cheng, Chen and Li [2] proved that the problem is NP-hard, and presented a pseudo-polynomial time algorithm. This paper is an extension of the above work with three main contributions. First, we will resolve the special case $n_T = 0$ of the problem by presenting an optimal algorithm. Its time complexity is $O(n^2)$, and becomes $O(n \log n)$ if all $w_i$ are the same. This problem, denoted by $P_1$: $1|contr, n_T = 0|TCC$, models the following scenario. For some applications in logistics and supply chain management, jobs denote orders from customers, and the machine denotes the manufacturer. In some cases, while tardy jobs are allowed, having tardy jobs may damage the goodwill of the customers, so the manufacturer is concerned with the trade-off between $n_T$ and $TCC$. In other cases, customers will accept no tardy jobs. So the manufacturer must find a solution to minimize the compression cost with the constraint that $n_T = 0$. Note that while the algorithm given in [2] still requires pseudo-polynomial time for $n_T = 0$, we present a strongly polynomial time algorithm for this case.

Second, we consider the problem with a new objective. The objective is to construct the trade-off curve between $n_T$ and the bottleneck objective function $MCC = \max_{i=1,\cdots,n} w_i x_i$, i.e., the maximum compression cost. This problem, denoted by $P_2$: $1|contr, n_T \leq k|MCC$, models the following situation: Given that the resource is limited, the decision maker (scheduler) seeks to find a solution that balances the amount of resource used by each job under the constraint that the number of tardy jobs is no greater than a given nonnegative integer $k$. We will show that this problem can be solved in $O(n^2 \log W)$ time, where $W = \max_{i=1,\cdots,n} w_i u_i$.

Third, we consider the problems with discrete controllable processing times. Scheduling problems with discrete controllable processing times have been studied by Chen, Lu and Tang [1], De et al. [3], [4], and Skutella [10], which have many real-world applications. For this situation, the allowed compression amount $x_i$ of job $J_i$ is in a finite set, i.e., it must be one of some given discrete values, instead of any value in the interval $[0, u_i]$. Hence, we assume that for each $J_i \in J$, the actual processing time $p'_i = p_i - x_i$ has $l_i$ possible values $a_{i1} = p_i > a_{i2} > \cdots > a_{il_i}$. Let $w_{ij}$ be the compression cost for the processing time $a_{ij}, i = 1, \cdots, n, j = 1, \cdots, l_i$. It is assumed that $0 = w_{i1} < w_{i2} < \cdots < w_{il_i}$. This assumption is reasonable because to achieve smaller processing times requires more resources, hence incurring higher costs [1]. We show that both $P_3$ : $1|disc\ contr, n_T \leq k|TCC$ and $P_4$ : $1|disc\ contr, n_T = 0|TCC$ are NP-hard even for a very special case where all the jobs have the same due-date, all $l_i = 2$ and all unit-compression costs $w_{i2}/a_{i2}, i = 1, \cdots, n$, are the same (i.e., all the unit-compression costs are the same for all the jobs), and present pseudo-polynomial time algorithms based on dynamic programming for the general case of $P_3$ and $P_4$. Moreover, we will present a strongly polynomial time algorithm for the problem $P_5:\ 1|disc\ contr, n_T \le k|MCC$. The time complexity is $O(n^2 \log(nl))$, where $l = \max_{i=1,\cdots,n} l_i$.

This paper is organized as follows: Sections 2 and 3 consider the problems $P_1$ and $P_2$, respectively, Section 4 studies the problems $P_3$ and $P_4$, and Section 5 studies the problem $P_5$. Final remarks are presented in Section 6.

## 2 Strongly polynomial solvability of the problem $P_1$

**Definition 2.1** *For the problem $P_1$, a solution is called feasible if it satisfies $\Sigma_{i=1}^n U_i = 0$.*

**Lemma 2.2** *There exists an optimal solution such that all the jobs are scheduled in the "Earliest Due-Date" (EDD) order.*

*Proof.* It can be shown by applying the standard pairwise job interchange argument. □

Hence, in the remainder of this section, we assume that the jobs are re-indexed such that $d_1 \le d_2 \le \cdots \le d_n$. Since $u_i$ is the upper bound on the compressibility of job $J_i, i = 1, 2, \cdots, n$, we have

**Corollary 2.3** *The problem $P_1$ has a feasible solution satisfying the EDD order if and only if $\Sigma_{j=1}^i (p_j - u_j) \le d_i, i = 1, 2, \cdots, n$.*

For each job $J_i, i = 1, 2, \cdots, n$, let the actual compression be $x_i$. Then job $J_i$ is not tardy if and only if $\Sigma_{j=1}^i (p_j - x_j) \le d_i$, i.e., $\Sigma_{j=1}^i x_j \ge \Sigma_{j=1}^i p_j - d_i$. Thus, the problem $P_1$ can be formulated as the following linear program:

$$\begin{aligned} \min \quad & \Sigma_{i=1}^n w_i x_i \\ s.t. \quad & \Sigma_{j=1}^i x_j \ge \Sigma_{j=1}^i p_j - d_i, \quad i = 1, 2 \cdots, n, \\ & 0 \le x_i \le u_i, \qquad\qquad\quad i = 1, 2, \cdots, n. \end{aligned}$$

It is clear that the linear program can be solved in polynomial time, so can $P_1$. In the following, we present a strongly polynomial time algorithm for this problem.

Algorithm $A_1$:

1. Renumber all the jobs in the $EDD$ order such that $d_1 \le d_2 \le \cdots \le d_n$. Set $x_i = 0$, $u_i' = u_i, i = 1, 2, \cdots, n$, and $k = 1$.

2. Schedule the jobs in the order of $J_1, J_2, \cdots, J_n$ with the actual processing times $p_1 - x_1, p_2 - x_2, \cdots, p_n - x_n$, respectively. If there are no tardy jobs after processing all the jobs, then go to 4; otherwise let $J_{t_k}$ be the first tardy job with completion time $C_{t_k}$, go to 3.

3. Assume that the unit-compression weights of the job set $\{J_1, J_2, \cdots, J_{t_k}\}$ satisfy $w_{j_1} \leq w_{j_2} \leq \cdots \leq w_{j_{t_k}}$ (that is to say, $J_{j_i}$ is the job with the $i$-th smallest unit-compression cost in $\{J_1, J_2, \cdots, J_{t_k}\}$). Note that the current bounds on the compressibility of these jobs are $u'_1, u'_2, \cdots, u'_{t_k}$, respectively. Let $s = \min\{i | \Sigma_{p=1}^{i} u'_{j_p} > C_{t_k} - d_{t_k}, 1 \leq i \leq t_k\}$. Then for $i = j_1, j_2, \cdots, j_{s-1}$, set $x_i \leftarrow x_i + u'_i$ and $u'_i \leftarrow 0$; for $i = j_s$, set $x_i \leftarrow x_i + C_{t_k} - d_{t_k} - \Sigma_{p=1}^{s-1} u'_{j_p}$ and $u'_i \leftarrow u'_i - x_i$. Set $k \leftarrow k + 1$, go to 2.

4. Output $x_i, i = 1, 2, \cdots, n$, as the final compression of job $J_i$, and the objective value $\Sigma_{i=1}^{n} w_i x_i$, stop.

**Lemma 2.4** *Consider the following linear program:*

$$\begin{aligned} \min \quad & \Sigma_{i=1}^{t} w_i y_i \\ s.t. \quad & \Sigma_{i=1}^{t} y_i \geq L, \\ & 0 \leq y_i \leq l_i, \quad i = 1, 2, \cdots, t. \end{aligned} \tag{1}$$

*where $L$ is a positive number and $0 \leq w_{j_1} \leq w_{j_2} \leq \cdots \leq w_{j_t}$. Let $s = \min\{i | \Sigma_{p=1}^{i} l_{j_i} > L, 1 \leq j \leq t\}$. Then*

$$y_{j_i} = \begin{cases} l_{j_i}, & \text{if } 1 \leq i \leq s-1, \\ L - \Sigma_{i=1}^{s-1} l_{j_i}, & \text{if } i = s, \\ 0, & \text{if } s+1 \leq i \leq t \end{cases}$$

*is an optimal solution.*

*Proof.* It is a continuous relaxation of the minimization version of a special bounded Knapsack problem [8], hence the lemma holds. □

**Theorem 2.5** *Algorithm $A_1$ produces an optimal solution to the problem $P_1$, and runs in time $O(n^2)$.*

*Proof.* First, it is clear that the solution produced by algorithm $A_1$ is feasible. We next prove its optimality.

Let $x_i^k$ and $x_i^\circ$ denote the compression of job $J_i, i = 1, 2, \cdots, n$, right after the $k$-th iteration of the algorithm, and the compression of job $J_i$ in an optimal solution satisfying the $EDD$ order, respectively.

We prove by induction on $k$ that $\Sigma_{i=1}^{t_k} w_i x_i^\circ \geq \Sigma_{i=1}^{t_k} w_i x_i^k$, and $\Sigma_{i=1}^{t_k} x_i^k$ is exactly the minimum possible total compression to assure no jobs in $\{J_1, J_2, \cdots, J_{t_k}\}$ are tardy, and thus $\Sigma_{i=1}^{t_k} x_i^\circ \geq \Sigma_{i=1}^{t_k} x_i^k$ for every $k \geq 1$. It states that the algorithm yields a solution that has the objective value and the total compression no greater than those of the optimal solution, and thus is optimal.

For $k = 1$, in order to guarantee that job $J_{t_1}$ is not tardy, $\Sigma_{i=1}^{t_1} x_i \geq C_{t_1} - d_{t_1}$ must hold. From Lemma 2.4, we conclude that $x_1^1, x_2^1, \cdots, x_{t_1}^1$ is an optimal solution to the following linear program:

$$\begin{aligned} \min \quad & \Sigma_{i=1}^{t_1} w_i y_i \\ s.t. \quad & \Sigma_{i=1}^{t_1} y_i \geq C_{t_1} - d_{t_1}, \\ & 0 \leq y_i \leq u_i, \qquad i = 1, 2, \cdots, t_1. \end{aligned} \tag{2}$$

On the other hand, $x_1^\circ, x_2^\circ, \cdots, x_{t_1}^\circ$ is a feasible solution to (2), hence $\Sigma_{i=1}^{t_1} w_i x_i^\circ \geq \Sigma_{i=1}^{t_1} w_i x_i^1$. Because $\Sigma_{i=1}^{t_1} x_i^1 = C_{t_1} - d_{t_1}$, we know that $\Sigma_{i=1}^{t_1} x_i^1$ is the minimum possible total compression such that job $J_{t_1}$ is not tardy. Hence, the result is true for $k = 1$.

In general, suppose that the result is true for $k-1$, that is, $\Sigma_{i=1}^{t_{k-1}} w_i x_i^\circ \geq \Sigma_{i=1}^{t_{k-1}} w_i x_i^{k-1}$ and $\Sigma_{i=1}^{t_{k-1}} x_i^\circ \geq \Sigma_{i=1}^{t_{k-1}} x_i^{k-1}$, and no jobs in $\{J_1, J_2, \cdots, J_{t_{k-1}}\}$ are tardy with the processing times $p_i - x_i^{k-1}, i = 1, 2, \cdots, t_{k-1}$.

Since $\Sigma_{i=1}^{t_{k-1}} x_i^{k-1}$ is the minimum possible total compression that guarantees that no jobs in $\{J_1, J_2, \cdots, J_{t_{k-1}}\}$ are tardy, to make job $J_{t_k}$ not tardy, the new compression must be at least $C_{t_k} - d_{t_k}$. From the algorithm, we know that $\Sigma_{i=1}^{t_k} x_i^k = \Sigma_{i=1}^{t_{k-1}} x_i^{k-1} + C_{t_k} - d_{t_k}$. It implies that $\Sigma_{i=1}^{t_k} x_i^k$ is exactly the minimum possible total compression to ensure that no jobs in $\{J_1, J_2, \cdots, J_{t_k}\}$ are tardy, and thus

$$\Sigma_{i=1}^{t_k} x_i^\circ \geq \Sigma_{i=1}^{t_k} x_i^k. \tag{3}$$

To show that $\Sigma_{i=1}^{t_k} w_i x_i^\circ \geq \Sigma_{i=1}^{t_k} w_i x_i^k$, we divide the compressions $x_i^\circ, i = 1, 2, \cdots, t_k$ into two parts $x_i'$ and $x_i''$, and prove the result by verifying that the total cost of the first parts of all jobs is no less than that of the compressions $x_1^{k-1}, x_2^{k-1}, \cdots, x_{t_{k-1}}^{k-1}$, and the cost of the second parts of all jobs is no less than that of the new compressions in the $k$-th iteration $\triangle x_1^k \doteq x_1^k - x_1^{k-1}, \triangle x_2^k \doteq x_2^k - x_2^{k-1}, \cdots, \triangle x_{t_k}^k \doteq x_{t_k}^k - x_{t_k}^{k-1}$. First, $x_i', i = 1, 2, \cdots, t_{k-1}$ are determined such that:

i) If $x_i^\circ \geq x_i^{k-1}$, then $x_i^\circ \geq x_i' \geq x_i^{k-1}$, if $x_i^\circ < x_i^{k-1}$, then $x_i' = x_i^\circ$.

ii) If the processing time of job $J_i, i = 1, 2, \cdots, t_{k-1}$ is $p_i' = p_i - x_i'$, then the jobs $J_1, J_2, \cdots, J_{t_{k-1}}$ are all not tardy by the $EDD$ rule, and

$$\Sigma_{i=1}^{t_{k-1}} x_i' = \Sigma_{i=1}^{t_{k-1}} x_i^{k-1}. \tag{4}$$

Furthermore, let $x_i' = 0, i = t_{k-1}+1, \cdots, t_k$. Thus, let the second part of $x_i^\circ$ be $x_i'' = x_i^\circ - x_i', i = 1, 2, \cdots, t_k$.

In fact, this construction can be performed as follows: Let $z_i = \min\{x_i^\circ, x_i^{k-1}\}, i = 1, 2, \cdots, t_{k-1}$ and $T = \Sigma_{i=1}^{t_{k-1}} x_i^{k-1} - \Sigma_{i=1}^{t_{k-1}} z_i$. Let $v = \min\{i | \Sigma_{j=1}^{i} (x_j^\circ - z_j) > T, 1 \leq i \leq t_{k-1}\}$. We then define $x_i', i = 1, 2, \cdots, t_{k-1}$ as follows (note that $x_i' = 0$ as above, $i = t_{k-1}+1, \cdots, t_k$):

$$x_i' = \begin{cases} x_i^\circ, & \text{for} \quad 1 \leq i \leq v-1, \\ z_i + T - \Sigma_{i=1}^{v-1}(x_i^\circ - z_i), & \text{for} \quad i = v, \\ z_i, & \text{for} \quad v < i \leq t_{k-1}. \end{cases} \tag{5}$$

Obviously, $x_i', i = 1, 2, \cdots, t_{k-1}$ satisfy i) and (4). Hence, we only need to prove that no jobs in $\{J_1, J_2, \cdots, J_{t_{k-1}}\}$ with processing times $p_i' = p_i - x_i'$ are tardy. We show this result by contradiction. Suppose that there is a tardy job. From the algorithm, we know that job $J_{t_{k-1}}$ is an on-time job after the $k-1$-th iteration in the algorithm with processing times $p_i - x_i^{k-1}, i = 1, \cdots, t_{k-1}$, thus

$$d_{t_{k-1}} = \Sigma_{i=1}^{t_{k-1}} (p_i - x_i^{k-1}). \tag{6}$$

Since $\Sigma_{i=1}^{t_{k-1}} x_i' = \Sigma_{i=1}^{t_{k-1}} x_i^{k-1}$, (6) implies $d_{t_{k-1}} = \Sigma_{i=1}^{t_{k-1}} (p_i - x_i')$, and $J_{t_{k-1}}$ is not tardy with the processing times $p_i', i = 1, \cdots, t_{k-1}$. Thus, let $J_{t_j}, j \leq k-2$ be the first tardy job according to the order from $J_{t_{k-1}}$ to $J_1$ with processing times $p_i', i = 1, \cdots, t_{k-1}$. If $t_j < v$, then from the construction of $x_i'$ we conclude that $\Sigma_{i=1}^{t_j} (p_i - x_i') = \Sigma_{i=1}^{t_j} (p_i - x_i^\circ) \leq d_{t_j}$, which implies that $J_{t_j}$ is not tardy, a contradiction. Thus, $t_j \geq v$.

From the algorithm, we know that job $J_{t_j}$ is an on-time job right after the $j$-th iteration with the processing times $p_i - x_i^j, i = 1, \cdots, t_j$, thus $d_{t_j} = \Sigma_{i=1}^{t_j} (p_i - x_i^j)$. Hence, combining this with (6), we have

$$\begin{aligned} d_{t_{k-1}} - d_{t_j} &= \Sigma_{i=1}^{t_{k-1}} (p_i - x_i^{k-1}) - \Sigma_{i=1}^{t_j} (p_i - x_i^j) \\ &= \Sigma_{i=t_j+1}^{t_{k-1}} p_i - \Sigma_{i=1}^{t_j} (x_i^{k-1} - x_i^j) - \Sigma_{i=t_j+1}^{t_{k-1}} x_i^{k-1} \\ &\leq \Sigma_{i=t_j+1}^{t_{k-1}} p_i - \Sigma_{i=t_j+1}^{t_{k-1}} x_i^{k-1} \quad (\text{since } j < k-1 \text{ implies } x_i^{k-1} - x_i^j \geq 0\ ) \\ &\leq \Sigma_{i=t_j+1}^{t_{k-1}} p_i - \Sigma_{i=t_j+1}^{t_{k-1}} z_i \quad (\text{since } z_i = \min\{x_i^\circ, x_i^{k-1}\} \leq x_i^{k-1}, i = 1, 2, \cdots, t_{k-1}) \\ &= \Sigma_{i=t_j+1}^{t_{k-1}} p_i - \Sigma_{i=t_j+1}^{t_{k-1}} x_i'. \quad (\text{by (5)}) \end{aligned} \tag{7}$$

Because of the assumption that job $J_{t_j}$ is tardy with the processing times $p'_i, i = 1, \cdots, t_{k-1}$, we have $\Sigma_{i=1}^{t_j}(p_i - x'_i) > d_{t_j}$. Combining this with (7), we obtain $\Sigma_{i=1}^{t_k}(p_i - x'_i) > d_{t_j} + \Sigma_{i=t_j+1}^{t_{k-1}}(p_i - x'_i) > d_{t_{k-1}}$. Thus $J_{t_{k-1}}$ is tardy with the processing times $p'_i, i = 1, \cdots, t_{k-1}$, a contradiction. Therefore, we conclude that no jobs in $\{J_1, J_2, \cdots, J_{t_{k-1}}\}$ are tardy.

Now we return to the proof of the theorem. On the one hand, the compressions $x'_1, x'_2, \cdots, x'_{t_{k-1}}$ make the jobs $J_1, J_2, \cdots, J_{t_{k-1}}$ not tardy; hence, from the induction assumption, we know that

$$\Sigma_{i=1}^{t_{k-1}} w_i x'_i \geq \Sigma_{i=1}^{t_{k-1}} w_i^{k-1} x_i^{k-1}. \tag{8}$$

On the other hand, subtracting (4) from (3), we get $\Sigma_{i=1}^{t_k} x''_i \geq \Sigma_{i=1}^{t_k} \triangle x_i^k = C_{t_k} - d_{t_k}$. Because $x'_i$ satisfies i), we have

$$0 \leq x''_i = x_i^\circ - x'_i \leq u_i - x'_i \leq u_i - x_i^{k-1}, i = 1, 2, \cdots, t_{k-1}.$$

By definition, we know

$$0 \leq x''_i = x_i^\circ \leq u_i, i = t_{k-1} + 1, \cdots, t_k.$$

Therefore, $x''_i, i = 1, 2, \cdots, t_k$, is a feasible solution to the following linear program:

$$\begin{array}{lll} \min & \Sigma_{i=1}^{t_k} w_i y_i & \\ & & \\ s.t. & \Sigma_{i=1}^{t_k} y_i \geq C_{t_k} - d_{t_k}, & \\ & & \\ & 0 \leq y_i \leq u_i - x_i^{k-1}, & i = 1, 2, \cdots, t_{k-1}, \\ & & \\ & 0 \leq y_i \leq u_i, & i = t_{k-1} + 1, \cdots t_k. \end{array} \tag{9}$$

From Lemma 2.4 and the algorithm, we conclude that $\triangle x_i^k, i = 1, 2, \cdots, t_k$ is an optimal solution to the above linear program (9). Thus, we obtain

$$\Sigma_{i=1}^{t_k} w_i \triangle x_i^k \leq \Sigma_{i=1}^{t_k} w_i x''_i. \tag{10}$$

Adding (8) and (10), we get $\Sigma_{i=1}^{t_k} w_i x_i^k \leq \Sigma_{i=1}^{t_k} w_i x_i^\circ$. Thus, the solution produced by algorithm $A_1$ is an optimal solution to the problem $P_1$.

We next study the time complexity of algorithm $A_1$. It is clear that Step 1 takes $O(n \log n)$ time. Steps 2-3 may iterate at most $n$ times and each iteration takes $O(n)$ time. Hence, algorithm $A_1$ runs in $O(n^2)$ in the worst case and is a strongly polynomial algorithm. $\Box$

If all unit-compression costs are the same, i.e., $w_i = w, i = 1, \cdots, n$, it can be shown similarly that the following simplified version of $A_1$ yields an optimal solution in time $O(n \log n)$.

Algorithm $A'_1$:

1. Renumber the jobs in the $EDD$ order such that $d_1 \le d_2 \le \cdots \le d_n$.
2. Compute $r$ such that $\Sigma_{i=1}^{r} p_i - d_r = \max\{\Sigma_{i=1}^{j} p_i - d_j | j = 1, 2, \cdots, n\}$.
3. Let $s = \min\{j | \Sigma_{i=1}^{j} u_i > \Sigma_{i=1}^{r} p_i - d_r, 1 \le j \le n\}$. Then, for $1 \le i \le s-1$, let $x_i = u_i$; for $i = s$, let $x_i = \Sigma_{i=1}^{r} p_i - d_r - \Sigma_{i=1}^{s-1} u_i$; and for $s+1 \le i \le n$, let $x_i = 0$.
4. Output $x_1, x_2, \cdots, x_n$ and $\Sigma_{i=1}^{n} w_i x_i$, stop.

# 3 Polynomial solvability of the problem $P_2$

To solve the problem $P_2$, we assume that all $w_i x_i, i = 1, 2, \cdots, n$ are integers, thus the objective value $\max\{w_i x_i, i = 1, 2, \cdots, n\}$ is also an integer. This assumption should not be a serious limitation [2], since the amount of allocation can always be expressed in the smallest unit of resource for all practical purposes, which makes the costs integers. Let $\max\{w_i u_i, i = 1, 2, \cdots, n\} = W$.

It is well-known that Moore's algorithm can solve the problem $1||n_T$ in $O(n^2)$ time [7]. In the following we present an optimal algorithm for $P_2$ by combining Moore's algorithm with the bisection method.

**Definition 3.1** *Consider an instance of the problem $1||n_T$ with the processing times $\{p'_i \mid i = 1, 2, \cdots, n\}$. If its optimal objective value is $n_T \le k$, then we say that $\{p'_i \mid i = 1, 2, \cdots, n\}$ is a feasible solution to the problem $P_2$. The problem $P_2$ is called feasible if it has at least one feasible solution.*

In the remainder of this section, we use $p^t$ to denote the set consisting of processing times $p_i - \min\{t/w_i, u_i\}, i = 1, 2, \cdots, n$, where $p_i$ is the initial processing time of job $J_i$. Specifically, $p^0 = \{p'_i = p_i \mid i = 1, 2, \cdots, n\}$ and $p^W = \{p'_i = p_i - u_i \mid i = 1, 2, \cdots, n\}$. The following result is trivial.

**Lemma 3.2** *(1) Let $\{p'_i \mid i = 1, 2, \cdots, n\}$ be a feasible solution to the problem $P_2$. If $p''_i \le p'_i, i = 1, 2, \cdots, n$, then $\{p''_i \mid i = 1, 2, \cdots, n\}$ is a feasible solution to the problem $P_2$, too. (2) The problem $P_2$ has a feasible solution if and only if $p^W$ is a feasible solution.*

Algorithm $A_2$:

1. Invoke Moore's algorithm to solve the instance of $1||n_T$ with actual processing times $p^0$. If $n_T \leq k$, then the current solution is optimal with the objective value $MCC = 0$, stop; otherwise, go to 2.

2. Invoke Moore's algorithm to solve the instance of $1||n_T$ with actual processing times $p^W$. If $n_T > k$, then output that the problem $P_2$ has no feasible solution, stop; Otherwise, let $t = W$ and $t' = 0$.

3. If $t - t' = 1$, then output that the solution $p^t$ is optimal with the objective value $MCC = t$, stop; otherwise set $l = \lceil (t + t')/2 \rceil$, and invoke Moore's algorithm to solve the instance of $1||n_T$ with actual processing times $p^l$. If $n_T \leq k$, then set $t = l$ and go back to 3; otherwise, set $t' = l$ and go to 3.

**Theorem 3.3** *If the problem $P_2$ is feasible, then the solution obtained by algorithm $A_2$ is an optimal solution, and the time complexity of algorithm $A_2$ is $O(n^2 \log W)$.*

*Proof.* If the algorithm stops at Step 2, then $p^W$ is not a feasible solution, and hence by Lemma 3.2(2), the problem is infeasible.

We now consider the case that the problem $P_2$ is feasible. If the algorithm stops at Step 1, then a solution $p^0$ is obtained with the objective value 0, which is trivially an optimal solution. Hence, we suppose in the following that the optimal objective value is not 0. Then we can claim that $p^W$ is a feasible solution, whereas $p^0$ is not. So, by the bisection procedure, algorithm $A_2$ can get $t$ and $t'$ such that $t - t' = 1$, $p^t$ is feasible, but $p^{t'} = p^{t-1}$ is not. Thus algorithm $A_2$ outputs a feasible solution $p^t$ with the objective value $MCC = t$ when it stops.

Let $\{p'_i \mid i = 1, 2, \cdots, n\}$ be any feasible solution of the problem $P_2$ with the objective value $MCC' = \max\{(p_i - p'_i)w_i \mid i = 1, 2, \cdots, n\}$. We now prove by contradiction that $MCC = t \leq MCC'$ and hence $p^t$ must be the optimal solution. Suppose $MCC > MCC'$. As there is no integer between $t-1$ and $t$, $MCC > MCC'$ implies that $t - 1 \geq MCC'$. So, by Lemma 3.3(1), $p^{t-1}$ is also feasible, However $t' = t - 1$, and we know that $p^{t'}$ is not feasible, a contradiction. Thus, $MCC > MCC'$ is not true and $p^t$ is an optimal solution to the problem $P_2$.

Because Step 3 may repeat for at most $\log W$ times and the time complexity of Moore's algorithm is $O(n^2)$, we conclude that the time complexity of algorithm $A_2$ is $O(n^2 \log W)$. □

## 4 NP-hardness of the problems $P_3$ and $P_4$

Cheng, Chen and Li [2] proved that the problem $P_0 : 1 \mid contr, n_T \leq k \mid TCC$ is NP-hard. Now we discuss the problem $P_3$ where the possible compressions of each job are some discrete values.

**Theorem 4.1** *The problem $P_3$ is $NP-$hard even if all the jobs have the same due-date, all $l_i = 2$, and all the unit-compression costs are the same.*

*Proof.* Since all the unit-compression costs are the same, the objective value of the problem $P_3$ is equivalent to the total compression.

We show the result by reducing the Partition problem [6] to this problem. Given an instance $I$ of the Partition problem with a set of positive integers $\{h_1, h_2 \cdots, h_n\}$ and $2B = \Sigma_{i=1}^{n} h_i$, we construct an instance $II$ of the problem $P_4$ as follows: Associated with each $h_i$, $i = 1, \cdots, n$, is job $J_i$ with

$$p_i = h_i, \quad p_i' = p_i - x_i \in \{h_i, 0\}, \quad d_i = B, \quad i = 1, 2, \cdots, n.$$

In addition, we construct $n$ more jobs $J_{n+1}, \cdots, J_{2n}$ with

$$p_i = 2B, \quad p_i' = p_i - x_i \in \{2B, 0\}, \quad d_i = B, \quad i = n+1, n+2, \cdots, 2n.$$

Define $k = n$ and threshold $UB = B$. We prove that instance $I$ has a solution if and only if instance $II$ has a solution with $n_T \leq k$ and the objective is no greater than $UB$.

If $I$ has a solution, then there exist two subsets $H_1$ and $H_2$ of $H$ such that $H_1 \cup H_2 = H$, $H_1 \cap H_2 = \emptyset$ and $\sum_{h_i \in H_1} h_i = \sum_{h_i \in H_2} h_i = B$. Construct a solution for instance $II$ as follows: Let the compression of $J_i, i \in H_1$, be $x_i = h_i$ for each $i \in H_1$, and $x_i = 0$ for all other jobs. Then the first $n$ jobs can be completed early or on-time while the last $n$ jobs are tardy. Hence, the number of tardy jobs is $k = n$, and the total compression is exactly $B$.

Next, suppose $II$ has a solution such that $n_T \leq k = n$ and the objective value (i.e., the total compression) is no greater than $UB$. If a job $J_i, n+1 \leq i \leq 2n$, is compressed, then the total compression is at least $2B > UB$, a contradiction. Thus, none of the last $n$ jobs can be compressed, and all of them are tardy. That is to say, the first $n$ jobs must be completed early or on time. If the total compression of the jobs $J_1, J_2, \cdots, J_n$ is less than $B$, then their total actual processing time is more than $2B - B = B$, hence there exists at least a tardy job, a contradiction. Since $UB = B$, the total compression of the jobs $J_1, J_2, \cdots, J_n$ is exactly $B$. It follows that there exists a subset $H \subseteq \{1, 2, \cdots, n\}$ such that $\Sigma_{i \in H} a_i = B$. □

We have shown that $1|contr, n_T = 0|TCC$ is strongly polynomial time solvable. However, the discrete controllable case becomes $NP$-hard.

**Theorem 4.2** *The problem $P_4$ is $NP-$hard even if all the jobs have the same due-date, all $l_i = 2$, and all the unit-compression costs are the same.*

*Proof.* Similarly, all the unit-compression costs being the same, the objective value of the problem $P_4$ is equivalent to the total compression.

We again show the result by reducing the Partition problem to this problem. Given an instance $I$ of the Partition problem with a set of positive integers $\{h_1, h_2 \cdots, h_n\}$ and $2B = \Sigma_{i=1}^{n} h_i$, we construct an instance $II$ of the problem $P_3$ as follows: Associated with each $h_i$, $i = 1, \cdots, n$, is job $J_i$ with

$$p_i = 2h_i, \quad p_i' = p_i - x_i \in \{2h_i, 3h_i/2\}, \quad d_i = 7B/2, \quad i = 1, 2, \cdots, n.$$

Define the threshold $UB = B/2$. It can easily be shown that instance $I$ has a solution if and only if instance $II$ has a solution with $n_T = 0$ and the objective value is no greater than $UB$. $\square$

Next we present a pseudo-polynomial time algorithm that solves the general case of the problem $P_3$ optimally. Note that if the problem $P_3$ is feasible (i.e., the number of tardy jobs is no greater than $k$), there must exist an optimal solution such that the early and on-time jobs are scheduled in the $EDD$ order, and the tardy jobs are scheduled in any order following all the early and on-time jobs. Furthermore, the tardy jobs are not compressed since we are to minimize the compression cost. Hence, we assume that $d_1 \leq d_2 \leq \cdots \leq d_n$.

Algorithm $A_3$:

1. Let $f(i, t, q)$ be the minimum total cost of a partial solution containing the first $i$ jobs, $J_1, J_2, \cdots, J_i$, given that the completion time of the early and on-time jobs in this partial solution is exactly $t$, and the number of tardy jobs is exactly $q$ $(1 \leq i \leq n, 0 \leq t \leq d_n, 0 \leq q \leq k)$.

2. Recursive relations: For $i = 2, 3, \cdots, n$, $t = 0, 1, \cdots, d_n$ and $q = 0, 1, \cdots, k$:

$$f(i,t,q) = \begin{cases} \min\{f(i-1,t,q-1), \min\{f(i-1,t-a_{ij},q) + w_{ij}, i = 1,2,\cdots,l_i\}\}, \\ \qquad\qquad\qquad \text{if} \quad t \leq d_{i-1}, \\ \min\{f(i-1,t-a_{ij},q) + w_{ij}, i = 1,2,\cdots,l_i\}, \\ \qquad\qquad\qquad \text{if} \quad d_{i-1} < t \leq d_i, \\ \infty, \\ \qquad\qquad\qquad \text{if} \quad t > d_i. \end{cases}$$

$$x_i = \begin{cases} p_i - a_{ij}, & \text{if} \quad f(i,t,q) = f(i-1,t-a_{ij},q) + w_{ij}, \\ 0, & \text{otherwise.} \end{cases}$$

3. Initial values: For $t = 0, 1, \cdots, d_n$:

$$f(1,t,0) = \begin{cases} w_{1j}, & \text{if} \quad d_1 \geq t = a_{1j}, \\ \infty, & \text{otherwise}, \end{cases}$$

$$f(1,t,1) = 0, \qquad t = 0, 1, \cdots, d_n.$$

$$x_1 = \begin{cases} p_1 - a_{ij}, & \text{if} \quad f(1,t,0) = w_{1j}, \\ 0, & \text{otherwise}. \end{cases}$$

4. An optimal solution can be obtained by computing

$$\min\{f(n,t,0), f(n,t,1) \cdots, f(n,t,k) | t = 0, \cdots, d_n\}.$$

**Remark 4.3** *Let $l = \max_{i=1,2,\cdots,n} l_i$, the time complexity of algorithm $A_2$ is $O(nlkd_n + n\log n)$ since we try all possible values of $i$ $(i = 1, \cdots, n)$, all possible values of $t$ $(t = 0, 1, \cdots, d_n)$, and all possible values of $q$ $(q = 0, 1, \cdots, k)$, and the computation of $f(i,t,q)$ needs $O(l)$ time for each possible state $(i,t,q)$. In addition, sorting jobs in non-decreasing order of due-date takes $O(n\log n)$ time. Therefore, algorithm $A_3$ is pseudo-polynomial time, implying that the problem $P_3$ is only NP-hard in the ordinary sense.*

**Remark 4.4** *Algorithm $A_3$ can also be used to solve $P_4$ by keeping $q = k = 0$. Hence, the time complexity of the algorithm becomes $O(nld_n + n\log n)$, and $P_4$ is NP-hard in the ordinary sense.*

# 5 Strongly polynomial solvability of the problem $P_5$

The main idea of the algorithm for $P_5$ is similar to that for $P_2$. However, we can construct a strongly polynomial time algorithm by making minor modifications. Before we present the algorithm, let $w_0 = 0$, and re-arrange the compression costs of all the jobs in non-increasing order of their values. Then we express their different values as follows: $0 = w_0 < w_1 < w_2 < \cdots < w_L$.

**Definition 5.1** *For a given positive number $S$, define $j_i = \max\{j | w_{ij} \leq S, j = 1, 2, \cdots, l_i\}$ for every $i = 1, \cdots, n$, and let $p^S = \{p'_i = a_{ij_i} | i = 1, 2, \cdots, n\}$. Specifically, $p^{w_0} = \{p'_i = a_{i1} = p_i | i = 1, 2, \cdots, n\}$ and $p^{w_L} = \{p'_i = a_{i1} = p_i | i = 1, 2, \cdots, n\}$.*

Algorithm $A_4$:

1. Invoke Moore's algorithm to compute the objective value of the instance of $1||n_T$ with actual processing times $p^{w_0}$. If the objective value $n_T \leq k$, then output $MCC = 0$, stop; otherwise, go to 2.

2. Invoke Moore's algorithm to compute the objective value of the instance of $1||n_T$ with actual processing times $p^{w_L}$. If the objective value $n_T > k$, then output that the problem $P_5$ has no a feasible solution, stop; otherwise, let $t = L$ and $t' = 0$.

3. If $t - t' = 1$, then output that the solution $p^{w_t}$ is optimal with objective value $MCC = w_t$, stop; otherwise set $l = \lceil (t + t')/2 \rceil$, and invoke Moore's algorithm to compute the instance of $1||n_T$ with actual processing times $p^{w_l}$. If $n_T \leq k$, then set $t = l$ and go back to 3; otherwise, set $t' = l$ and go back to 3.

**Theorem 5.2** *If the problem $P_5$ has feasible solutions, then the solution obtained by algorithm $A_4$ is an optimal solution, and the time complexity of algorithm $A_4$ is $O(n^2 \log(nl))$, where $l = \max_{i=1,2,\cdots,n} l_i$.*

*Proof.* With an argument analogous to the proof of Theorem 3.3, we can show that the solution produced by algorithm $A_4$ is optimal. Since the compression costs of all the jobs have at most $L \leq nl$ different values, Step 3 iterates at most $L$ times. Therefore, we conclude that the time complexity of algorithm $A_4$ is $O(n^2 \log(nl))$. □

# 6 Conclusions

In this paper we considered single-machine scheduling with continuously and discretely controllable processing times. The goal is to construct the trade-off curve between the number of tardy jobs and the total or maximum compression cost. We found that the levels of difficulty between the sum objective (i.e., total compression cost) and bottleneck objective cases, and between the continuous and discrete models, are quiet different. For the sum objective case, the problems $1|contr, n_T \leq k|TCC$, $1|disc\ contr, n_T \leq k|TCC$ and $1|disc\ contr, n_T = 0|TCC$ are NP-hard, but for the bottleneck case, both the problems $1|contr, n_T \leq k|MCC$ and $1|disc\ contr, n_T \leq k|MCC$ are polynomial solvable. For the continuous model, the problem $1|contr, n_T = 0|TCC$ is strongly polynomial solvable, but for the discrete model, even the special case of the problem $1|disc\ contr, n_T = 0|TCC$ is NP-hard.

# References

[1] Z.-L. Chen, Q. Lu, G.C. Tang, Single machine scheduling with discretely controllable processing times, *Operations Research Letters*, 21(1997), 69-76.

[2] T.C.E. Cheng, Z.-L. Chen, C.-L. Li, Single-machine scheduling with trade-off between number of tardy jobs and resource allocation, *Operations Research Letters*, 19(1996), 237-242.

[3] P. De, E.J. Dunne, J.B. Ghosh, C.E. Wells, The discrete time-cost trade-off problem revisited, *European Journal of Operational Research*, 81(1995), 225-238.

[4] P. De, E.J. Dunne, J.B. Ghosh, C.E. Wells, Complexity of the discrete time-cost trade-off problem for project networks, *Operations Research*, 45(1997), 302-306.

[5] R.L. Daniels and R.K. Sarin, Single machine scheduling with controllable processing times and number of jobs tardy, *Operations Research*, 37(1989), 981-984.

[6] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-hardness*, Freeman, San Francisco, CA, 1978.

[7] J.M. Moore, An $n$ job, one-machine sequencing algorithm for minimizing the number of late jobs, *Management Science*, 15(1968), 102-109.

[8] S. Martello, P. Toth, *Knapsack Problems: Algorithm and Computer Implementations*, John Wiley & Sons, Chichester, 1990.

[9] E. Nowicki, S. Zdrzalka, A survey of results for sequencing problems with controllable processing times, *Discrete Applied Mathematics*, 25(1990), 271-287.

[10] M. Skutella, Approximation algorithms for the discrete time-cost trade-off problem, *Mathematics of Operations Research*, 23(1998), 909-929.

[11] R.G. Vickson, Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine, *Operations Research*, 28(1980), 1155-1167.

[12] R.G. Vickson, Two single machine sequencing problems involving controllable job processing times, *IIE Transactions*, 12(1980), 258-262.

[13] L.N. Van Wassenhove, K.R. Baker, A bicriterion approach to time/cost trade-offs in sequencing, *European Journal of Operational Research*, 11(1982), 48-54.