Journal of Global Optimization (2005) 31:333–352 

# Minimizing Completion Time Variance with Compressible Processing Times

C.T. NG[1], X. CAI[2], T.C.E. CHENG[1] and S.S. LAM[3]
[1]*Department of Logistics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, P.R. China*
[2]*Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, P.R. China*
[3]*School of Business and Administration, The Open University of Hong Kong, Homantin, Hong Kong, P.R. China*

(Received 30 March 2004; accepted in revised form 1 April 2004)

**Abstract.** We introduce a new formulation of the standard completion time variance (CTV) problem with $n$ jobs and one machine, in which the job sequence and the processing times of the jobs are all decision variables. The processing time of job $i$ $(i = 1, \ldots, n)$ can be compressed by an amount within $[l_i, u_i]$, which however will incur a compression cost. The compression cost is a general convex non-decreasing function of the amount of the job processing time compressed. The objective is to minimize a weighted combination of the completion time variance and the total compression cost. We show that, under an agreeable condition on the bounds of the processing time compressions, a pseudo-polynomial algorithm can be derived to find an optimal solution for the problem. Based on the pseudo-polynomial time algorithm, two heuristic algorithms H1 and H2 are proposed for the general problem. The relative errors of both heuristic algorithms are guaranteed to be no more than $\delta$, where $\delta$ is a measure of deviation from the agreeable condition. While H1 can find an optimal solution for the agreeable problem, H2 is dominant for solving the general problem. We also derive a tight lower bound for the optimal solution of the general problem. The performance of H2 is evaluated by complete enumeration for small $n$, and by comparison with this tight lower bound for large $n$. Computational results (with $n$ up to 80) are reported, which show that the heuristic algorithm H2 in general can efficiently yield near optimal solutions, when $n$ is large.

**Key words:** Completion time variance, Compressible processing times, Scheduling

## 1. Introduction

The single-machine completion time variance (CTV) scheduling problem is well-known to be NP-hard in the ordinary sense. Its objective is to determine a job schedule so as to minimize the variance of job completion times. Originally, this problem was formulated by Merten and Muller [8] in 1972, motivated by the need to provide uniform responses to users' requests for retrieving computer data files. The model was later demonstrated to be applicable to any situation where it is desirable to provide a uniform treatment to all the jobs. Applications of the model may be found

in both services and manufacturing operations. For a review of the previous work on this problem, see De et al. [5] and Cai [2].

In this paper, we are concerned with an extension of the standard CTV problem to the situations where the processing time of a job can be compressed, if needed, at an extra cost. The compression of a job processing time in order to meet a certain due date occurs frequently in service and manufacturing environments. An optimal solution is to be sought which, on the one hand, yields the lowest possible completion time variance and, on the other hand, does not incur too much compression cost. Specifically, we formulate the problem as a model of determining simultaneously an optimal job sequence, as well as a set of optimal processing times, so that the weighted combination of the completion time variance and the total compression cost is minimized.

This is a new formulation of the CTV problem involving controllable processing times, although other models involving compressible processing times have been extensively investigated in the literature. In general, previous research usually considers linear scheduling costs. For example, Nowicki and Zdrzalka [11] addressed a problem of minimizing the maximum completion time plus the total compression cost. Panwalkar and Rajagopalan [13] investigated a problem with an objective function comprising earliness cost, tardiness cost and linear compression cost. Similarly, Alidaee and Ahmadian [1] considered the minimization of two types of scheduling cost: (A) the total compression cost plus the total flow time, and (B) the total compression cost plus total weighted earliness and weighted tardiness. Recently, Cheng et al., in [3] and [4], studied compressible processing times problems with convex compression costs. However, no research has considered a quadratic earliness/tardiness penalty function with a convex compression cost. For a comprehensive review of scheduling problems involving compressible processing times, see Nowicki and Zdrzalka [12].

The classical CTV problem, where the processing times are given, can be regarded as a special case of our model, since the processing times are in fact fixed when the lower bound and upper bound of the processing time of each job are equal. Since the classical CTV problem has already been proved to be NP-hard [7], our new problem is also NP-hard and it is thus unlikely that an algorithm that can find its optimal solution in polynomial time exists. In this paper, we show that, under a certain *agreeable condition* (which can be easily justified), a dynamic programming algorithm can be designed that can find an optimal solution for the problem in pseudo-polynomial time, bounded above by $O(nU(U-L+1)(U-L+n))$, where $U=\sum_{i=1}^{n} u_i, L=\sum_{i=1}^{n} l_i$, and $u_i$ and $l_i$ are the upper bound and the lower bound of the processing time of job $i$, respectively. Based on the pseudo-polynomial time algorithm, two heuristic algorithms H1 and H2

are proposed for the general problem, where the agreeable condition may not be satisfied. The relative errors of both heuristic algorithms are guaranteed to be no more than $\delta$, where $\delta$ is a measure of deviation from the agreeable condition. While H1 can find an optimal solution faster than H2 for the agreeable problem, H2 is dominant for solving the general problem. We also derive a tight lower bound for the optimal solution of the general problem. The performance of H2 is evaluated by complete enumeration for small $n$, and by comparison with this tight lower bound for large $n$. Computational results are reported, which show that the heuristic algorithm H2 in general can efficiently yield near optimal solutions, and demonstrate a trend of diminishing relative errors when $n$ grows.

The remainder of the paper is organized as follows. Section 2 gives a detailed formulation of the problem. Section 3 derives some basic results. A tight lower bound for the solution of the general problem will be given in Section 4. Section 5 develops the pseudo-polynomial algorithm, and analyzes its time complexity. In Section 6, the heuristic algorithms and their error analyses will be presented. Computational results are reported in Section 7. Finally, some concluding remarks are given in Section 8.

## 2. Problem Formulation

We consider the problem of scheduling $n$ independent and simultaneously available jobs on a single machine, which is assumed to be continuously available and can process only one job at a time. Job splitting and preemption are not allowed. Each job $i\,(i = 1, 2, \ldots, n)$ requires a positive integer processing time $p_i$, which can be chosen from an interval $[l_i, u_i]$, where $l_i$ and $u_i$ are positive integers and are called the lower bound and upper bound of the processing time of job $i$, respectively. As each processing time may be compressed from its upper bound, the amount of time compressed for job $i$ is thus $u_i - p_i$. The compression of job $i$ will incur a compression cost $f(u_i - p_i)$, where $f(x)$ is any *convex non-decreasing* function of $x \geqslant 0$. Let

$\Pi$ = the set of all permutations of the integers from 1 to $n$,
$\lambda = (\lambda(1), \ldots, \lambda(n)) \in \Pi$, a sequence that specifies job $\lambda(i)$ is processed in position $i, i = 1, \ldots, n$,
$C_i$ = the completion time of job $i$,
$\overline{C} = \frac{1}{n}\sum_{i=1}^{n} C_i$, the mean completion time,
$\mathbf{l} = (l_1 \ldots l_n)^t$, the vector of lower bounds,
$\mathbf{u} = (u_1 \ldots u_n)^t$, the vector of upper bounds.

The problem is to find an optimal solution so as to minimize a weighted linear combination of the completion time variance and the total compression cost. Specifically, we seek to determine a sequence $\lambda$ and a processing time vector $\mathbf{p} = (p_1 \ldots p_n)^t$ so as to minimize

$$CTVC(\lambda, \mathbf{p}) = \omega_0 \left[ \frac{1}{n} \sum_{i=1}^{n} (C_i - \overline{C})^2 \right] + \omega \sum_{i=1}^{n} f(u_i - p_i),$$

subject to $\lambda \in \Pi$ and $\mathbf{l} \leqslant \mathbf{p} \leqslant \mathbf{u}$, where $\omega_0 > 0$ is the weight associated with the completion time variance, and $\omega \geqslant 0$ is the weight associated with the total compression cost. Without loss of generality, we assume that $\omega_0 = n$. For convenience, we write: $CTV(\lambda, \mathbf{p}) = \sum_{i=1}^{n} (C_i - \overline{C})^2$ and $CC(\mathbf{p}) = \omega \sum_{i=1}^{n} f(u_i - p_i)$. Hence,

$$CTVC(\lambda, \mathbf{p}) = \sum_{i=1}^{n} (C_i - \overline{C})^2 + \omega \sum_{i=1}^{n} f(u_i - p_i) = CTV(\lambda, \mathbf{p}) + CC(\mathbf{p}).$$

It should be noted that $CC(\mathbf{p})$ is independent of the job sequence $\lambda$.

## 3. Definitions and Basic Results

The following two definitions will be used in this paper. The first one comes from the literature.

DEFINITION 1. A sequence is said to be *V-shaped with respect to the processing times* if the jobs preceding the shortest job (the job with the shortest processing time) are in the longest processing time (LPT) order whereas the jobs succeeding the shortest job are in the shortest processing time (SPT) order.

DEFINITION 2. A sequence is said to be *V-shaped with respect to the job indices* if the jobs preceding job 1 are sequenced in decreasing order of the job indices whereas the jobs succeeding job 1 are sequenced in increasing order of the job indices.

For example, if $p_1 = 4, p_2 = 3, p_3 = 2$, then the sequence (1,3,2) is V-shaped with respect to the processing times, but not V-shaped with respect to the job indices. On the other hand, the sequence (3,1,2) is V-shaped with respect to the job indices, but not V-shaped with respect to the processing times.

It is clear that if $p_1 \leqslant p_2 \cdots \leqslant p_n$, then any V-shaped sequence with respect to the processing times is also V-shaped with respect to the job indices, and vice versa.

The following lemma will be used for deriving Theorem 2.

LEMMA 1. *If $g$ is a convex function in an interval $I$, then, for any $x, y \in I$ with $x \leqslant y$, we have $g(x+z)+g(y-z) \leqslant g(x)+g(y)$ for all $z \in [0, y-x]$.*

*Proof.* Let $z = (y-x)\alpha$, where $\alpha \in [0,1]$. Then, $x+z = (1-\alpha)x + \alpha y$ and $y - z = \alpha x + (1-\alpha)y$. Since $g$ is convex, we have

$$g(x+z) \leqslant (1-\alpha)g(x) + \alpha g(y),$$

and

$$g(y-z) \leqslant \alpha g(x) + (1-\alpha)g(y).$$

Adding these two inequalities together will yield the result. □

We now examine the V-shaped property of an optimal sequence for the considered problem. For the classical CTV problem with fixed processing times, it has been known (cf. Eilon and Chowdhury [6]) that an optimal sequence must be V-shaped with respect to the processing times. Thus, if the optimal processing times of the $n$ jobs for our problem are given, the corresponding sequence that minimizes the $CTV$ part of our problem will be V-shaped with respect to the processing times. Since the $CC$ part is independent of the job sequence, the sequence that minimizes the overall $CTVC$ is clearly V-shaped with respect to the processing times. Therefore, we have the following theorem.

THEOREM 1. *If the $n$ optimal processing times are given, then there exists an optimal sequence which is V-shaped with respect to the optimal processing times.*

However, one should note that such a V-shaped sequence is entirely dependent upon the values of the optimal processing times. The difficulty is that, before the optimal processing times are found, we actually do not know which jobs will have the shortest optimal processing time, the second shortest optimal processing time, and so on. Theorem 2 below gives an answer to this question under the following agreeable condition.

Agreeable condition : $l_1 \leqslant l_2 \cdots \leqslant l_n$ and $u_1 \leqslant u_2 \leqslant \cdots \leqslant u_n$.

To justify this agreeable condition, we consider the following example. Suppose that a company can hire from 1 to $m$ workers to work for each of the $n$ jobs. That is, for two different jobs, the company may hire two different numbers of workers. For simplicity, the speeds of the $m$ workers are assumed to be the same. Let $u_i$ be the time taken for one worker to complete job $i$. We can re-label the job indices such that $u_1 \leqslant u_2 \leqslant \cdots \leqslant u_n$. Clearly, the minimum and maximum processing times to complete job $i$ are, respectively, corresponding to hiring $m$ and 1 worker(s) for job $i$. Thus, the lower bound and upper bound of the processing time for job $i$ are $l_i = u_i/m$ and $u_i$, respectively. It follows that $l_1 \leqslant l_2 \leqslant \cdots \leqslant l_n$.

THEOREM 2. *Under the agreeable condition, there exists an optimal processing time vector* $\mathbf{p}^* = (p_1^* \cdots p_n^*)^t$ *such that* $p_1^* \leqslant p_2^* \leqslant \cdots \leqslant p_n^*$.

*Proof.* Let $\mathbf{p}^\circ = (p_1^\circ \ldots p_n^\circ)^t$ be an optimal processing time vector and $\lambda^\circ$ be the corresponding optimal sequence. Suppose that $p_i^\circ > p_j^\circ$ for some $i < j$. Then, under the agreeable condition, we have

$$l_i \leqslant l_j \leqslant p_j^\circ < p_i^\circ \leqslant u_i \leqslant u_j,$$

which implies that $p_i^\circ \in [l_j, u_j]$ and $p_j^\circ \in [l_1, u_i]$. Let job $i$ and job $j$ be scheduled in position $i'$ and position $j'$ under $\lambda^\circ$, namely, $i = \lambda^\circ(i')$ and $j = \lambda^\circ(j')$. Consider the processing time vector $\mathbf{p}^\diamondsuit = (p_1^\diamondsuit \ldots p^\diamondsuit)^t$ and the sequence $\lambda^\diamondsuit \in \Pi$ such that, for $k = 1, 2, \ldots, n$,

$$p_k^\diamondsuit = \begin{cases} p_k^\circ, & k \neq i, j \\ p_j^\circ, & k = i \\ p_i^\circ, & k = j \end{cases}$$

$$\lambda_k^\diamondsuit = \begin{cases} \lambda^\circ(k), & k \neq i', j' \\ \lambda^\circ(j'), & k = i', \\ \lambda^\circ(i'), & k = j' \end{cases}$$

which means that $\mathbf{p}^\diamondsuit$ is obtained from $\mathbf{p}^\circ$ by interchanging the processing times of job $i$ and job $j$, and $\lambda^\diamondsuit$ is obtained from $\lambda^\circ$ by interchanging the positions of job $i$ and job $j$. It can easily be seen that $CTV(\lambda^\circ, \mathbf{p}^\circ) = CTV(\lambda^\diamondsuit, \mathbf{p}^\diamondsuit)$ However, for the $CC$ part, since $f(x)$ is convex non-decreasing, one has

$$\begin{aligned} CC(\mathbf{p}^\circ) - CC(\mathbf{p}^\diamondsuit) &= \omega\left\{\left[f(u_i - p_i^\circ) + f(u_j - p_j^\circ)\right] - \left[f(u_i - p_i^\diamondsuit) + f(u_j - p_j^\diamondsuit)\right]\right\} \\ &= \omega\left[f(u_i - p_i^\circ) + f(u_j - p_j^\circ) - f(u_i - p_j^\circ) - f(u_j - p_i^\circ)\right] \\ &\geqslant 0. \end{aligned}$$

The last inequality follows from Lemma 1 by putting $g = f, I = [0, \infty)$, $x = u_i - p_i^\circ, y = u_j - p_j^\circ$ and $z = p_i^\circ - p_j^\circ$. Hence, $CTVC(\lambda^\circ, \mathbf{p}^\circ) \geqslant CTVC(\lambda^\diamondsuit, \mathbf{p}^\diamondsuit)$. Since $CTVC(\lambda^\circ \mathbf{p}^\circ)$ is optimal, we have $CTVC(\lambda^\circ, \mathbf{p}^\circ) = CTVC(\lambda^\diamondsuit, \mathbf{p}^\diamondsuit)$.This means that if $p_i^\circ > p_j^\circ$ and $i < j$, then by interchanging the processing times and the job positions of job $i$ and job $j$, we can obtain a $CTVC$ value that is equal to the optimum. Thus, by interchanging the processing times and the job positions of job 1 and the job with the shortest processing time, and then interchanging the processing times and the job positions of job 2 and the job with the second shortest processing time, and so on, we will eventually obtain a set of $n$ optimal processing times for the $n$ jobs in non-decreasing order. □

Armed with Theorems 1 and 2, we can obtain Theorem 3.

THEOREM 3. *Under the agreeable condition, there exists an optimal sequence that is V-shaped with respect to the job indices.*

*Proof.* Theorem 2 shows that, under the agreeable condition, there exist $n$ optimal processing times such that $p_1^* \leqslant p_2^* \leqslant \cdots \leqslant p_n^*$. Since there exists an optimal sequence that is V-shaped with respect to these optimal processing times (Theorem 1), this sequence must also be V-shaped with respect to the job indices. □

## 4. A Tight Lower Bound

In this section, we will derive a tight lower bound for the optimal objective value of the general problem, where the agreeable condition may, or may not, be satisfied. Solutions obtained by the heuristic algorithm H2, to be proposed in Section 6, will be compared with this lower bound. This will result in an upper bound for the relative error of the heuristic algorithm. If the upper bound diminishes as $n$ grows, then the relative error of the heuristic algorithm must also diminish as $n$ grows. The following theorem provides such a tight lower bound. Here, we denote $l_{[i]}$ as the $i$-th smallest number in the set $\{l_1, \ldots, l_n\}$. Therefore, $l_{[1]} \leqslant \cdots \leqslant l_{[n]}$. Similarly, we denote $u_{(i)}$ as the $i$-th smallest number in the set $\{u_1, \ldots, u_n\}$, and we have $u_{(1)} \leqslant \cdots \leqslant u_{(n)}$. Let $\widehat{\mathbf{l}} = \left(l_{[1]} \ldots l_{[n]}\right)^t$ and $\widehat{\mathbf{u}} = \left(u_{(1)} \ldots u_{(n)}\right)^t$. For convenience, we denote the problem of minimizing $CTVC(\pi, \mathbf{p})$, subject to $\pi \in \Pi, \mathbf{l} \leqslant \mathbf{p} \leqslant \mathbf{u}$, by Pl, and the problem of minimizing $CTVC(\pi, \mathbf{p})$, subject to $\pi \in \Pi, \widehat{\mathbf{l}} \leqslant p \leqslant \widehat{\mathbf{u}}$, by P2.

THEOREM 4. *For any $l_1, \ldots, l_n$ and $u_1, \ldots, u_n$,*

$$\min_{\pi \in \Pi, \widehat{\mathbf{l}} \leqslant \mathbf{p} \leqslant \widehat{\mathbf{u}}} CTVC(\pi, \mathbf{p}) \leqslant \min_{\pi \in \Pi, \mathbf{l} \leqslant \mathbf{p} \leqslant \mathbf{u}} CTVC(\pi, \mathbf{p}).$$

*Proof.* Without loss of generality, we may assume that $u_1 \leqslant \cdots \leqslant u_n$. It is because we can re-label the jobs so that the lower bounds are in non-decreasing order. We have $\widehat{\mathbf{u}} = \mathbf{u}$. The remaining proof is similar to the proof of Theorem 2. Suppose that $l_i > l_j$ for some $i < j$. We consider the vector $\mathbf{l}^{\diamondsuit} = \left(l_1^{\diamondsuit} \ldots l_n^{\diamondsuit}\right)^t$ such that, for $k = 1, 2, \ldots, n$,

$$l_k^{\diamondsuit} = \begin{cases} l_k, & k \neq i, j \\ l_j, & k = i \\ l_i, & k = j \end{cases},$$

which means that $\mathbf{l}^{\diamondsuit}$ is obtained from $\mathbf{l}$ by interchanging the lower bounds of job $i$ and job $j$. Let $\mathbf{p}^{\circ} = (p_1^{\circ} \ldots p_n^{\circ})^t$ be an optimal processing time vector and $\lambda^{\circ}$ be the corresponding optimal sequence for Pl. Clearly, $\mathbf{l} \leqslant \mathbf{p}^{\circ} \leqslant \mathbf{u}$.

**Case 1:** $p_i^\circ \leqslant p_j^\circ$

By noting that

$$l_j < l_i \leqslant p_i^\circ \leqslant p_j^\circ,$$

we get $l_i^\diamondsuit = l_j < p_i^\circ$ and $l_j^\diamondsuit = l_i < p_j^\circ$. This implies that $\mathbf{l}^\diamondsuit \leqslant \mathbf{p}^\circ \leqslant \mathbf{u}$. Therefore,

$$\min_{x \in \Pi, \mathbf{l}^\diamondsuit \leqslant \mathbf{p} \leqslant \mathbf{u}} CTVC(\pi, \mathbf{p}) \leqslant CTVC(\lambda^\circ, \mathbf{p}^\circ) = \min_{\pi \in \Pi, \mathbf{l} \leqslant \mathbf{p} \leqslant \mathbf{u}} CTVC(\pi, \mathbf{p}).$$

**Case 2:** $p_i^\circ > p_j^\circ$

Let job $i$ and job $j$ be scheduled in position $i'$ and position $j'$ under $\lambda^\circ$, namely, $i = \lambda^\circ(i')$ and $j = \lambda^\circ(j')$. Consider the processing time vector $\mathbf{p}^\diamondsuit = (p_1^\diamondsuit \ldots p_n^\diamondsuit)^t$ and the sequence $\lambda^\diamondsuit \in \Pi$ such that, for $k = 1, 2, \ldots, n$,

$$p_k^\diamondsuit = \begin{cases} p_k^\circ, & k \neq i, j \\ p_j^\circ, & k = i \\ p_i^\circ, & k = j \end{cases}$$

and

$$\lambda^\diamondsuit(k) = \begin{cases} \lambda^\circ(k), & k \neq i', j' \\ \lambda^\circ(j'), & k = i' \\ \lambda^\circ(i'), & k = j' \end{cases},$$

which means that $\mathbf{p}^\diamondsuit$ is obtained from $\mathbf{p}^\circ$ by interchanging the processing times of job $i$ and job $j$, and $\lambda^\diamondsuit$ is obtained from $\lambda^\circ$ by interchanging the positions of job $i$ and job $j$. It can easily be seen that $CTV(\lambda^\circ, \mathbf{p}^\circ) = CTV(\lambda^\diamondsuit, \mathbf{p}^\diamondsuit)$. However, for the $CC$ part, since $f(x)$ is convex non-decreasing, one has

$$\begin{aligned} CC(\mathbf{p}^\circ) - CC(\mathbf{p}^\diamondsuit) &= \omega\left\{\left[f(u_i - p_i^\circ) + f\left(u_j - p_j^\circ\right)\right] - \left[f\left(u_i - p_i^\diamondsuit\right) + f\left(u_j - p_j^\diamondsuit\right)\right]\right\} \\ &= \omega\left[f(u_i - p_i^\circ) + f(u_j - p_j^\circ) - f(u_i - p_j^\circ) - f(u_j - p_i^\circ)\right] \\ &\geqslant 0. \end{aligned}$$

The last inequality follows from Lemma 1 by putting $g = f, I = [0, \infty)$, $x = u_i - p_i^\circ, y = u_j - p_j^\circ$ and $z = p_i^\circ - p_j^\circ$. Hence, $CTVC(\lambda^\circ, \mathbf{p}^\circ) \geqslant CTVC(\lambda^\diamondsuit, \mathbf{p}^\diamondsuit)$. By noting that

$$p_j^\circ < p_i^\circ \leqslant u_i \leqslant u_j,$$

we have $l_i^\diamondsuit = l_j \leqslant p_i^\diamondsuit = p_j^\circ < u_i$ and $l_j^\diamondsuit = l_i \leqslant p_j^\diamondsuit = p_i^\circ \leqslant u_j$. This implies that $\mathbf{l}^\diamondsuit \leqslant \mathbf{p}^\diamondsuit \leqslant \mathbf{u}$. Therefore,

$$\begin{aligned} \min_{\pi \in \Pi, \mathbf{l}^\diamondsuit \leqslant \mathbf{p} \leqslant \mathbf{u}} CTVC(\pi, \mathbf{p}) &\leqslant CTVC(\lambda^\diamondsuit, \mathbf{p}^\diamondsuit) \\ &\leqslant CTVC(\lambda^\circ, \mathbf{p}^\circ) = \min_{\pi \in \Pi, \mathbf{l} \leqslant \mathbf{p} \leqslant \mathbf{u}} CTVC(\pi, \mathbf{p}). \end{aligned}$$

This means that if $l_i > l_j$ for some $i < j$, then by interchanging the lower bounds of job $i$ and job $j$, we get a new $CTVC$ problem with an optimal objective value less than or equal to that of the original one. This operation of interchanging lower bounds is performed iteratively. Thus, by interchanging the lower bounds of job 1 and the job having the smallest lower bound, and then interchanging the lower bounds of job 2 and the job having the second smallest lower bound, and so on, we will eventually reach problem P2. □

Theorem 4 shows that the optimal objective value of P2 is a lower bound for the optimal objective value of P1. This lower bound is tight because, for each $n$, it is possible that the lower bounds and upper bounds of the jobs in P1 may satisfy the agreeable condition, and thus P1 is identical to P2. This concept of tight lower bound has been used in the literature, see [10] for example.

In the next section, we will design an algorithm that can find an optimal solution for any problem instance satisfying the agreeable condition. As P2 satisfies the agreeable condition, its optimal objective value can be found using this algorithm. In addition, this algorithm will be incorporated into the heuristic algorithms in Section 6 to find an approximate solution for the general problem.

## 5. A Solution Procedure for the Agreeable Problem

Theorem 3 shows that, under the agreeable condition, there exists an optimal sequence that is V-shaped with respect to the job indices. This suggests a dynamic programming approach to construct a pseudo-polynomial algorithm, which can obtain an optimal solution if the agreeable condition is satisfied. The design of the algorithm is an extension of the design in [2]. To begin with, we introduce the following auxiliary problem.

### 5.1. AN AUXILIARY PROBLEM

Let $C_i(\lambda, \mathbf{p}), i = 1, \ldots, n$, be the completion times corresponding to $\lambda \in \Pi$ and $\mathbf{p} \in P = \{(p_1 \ldots p_n)^t : p_i \in [l_i, u_i], i = 1, \ldots, n\}$, and $\bar{C}(\lambda, \mathbf{p}) = \frac{1}{n}\sum_{i=1}^{n} C_i(\lambda, \mathbf{p})$ be the mean completion time. To derive the algorithm, we introduce the following auxiliary problem:

$$\min_{d \in D, \lambda \in \Pi, p_i \in [l_i, u_i], i=1,\ldots,n} \left[ \gamma(\lambda, d, \mathbf{p}) = \sum_{j=1}^{n} (C_j(\lambda, \mathbf{p}) - d)^2 + \omega \sum_{j=1}^{n} f(u_j - p_j) \right], \tag{1}$$

where

$$D = \left\{1, 1+\frac{1}{n}, 1+\frac{2}{n}, \ldots, U-\frac{1}{n}, U\right\}$$

is defined as the feasible set of $d$ and $U = \sum_{i=1}^{n} u_i$ is the sum of all the upper bounds. It is easy to see that

$$\gamma(\lambda, d, \mathbf{p}) = \gamma(\lambda, \bar{C}(\lambda, \mathbf{p}), \mathbf{p}) + n(\bar{C}(\lambda, \mathbf{p}) - d)^2 \tag{2}$$

for any $d \in D, \lambda \in \Pi$ and $\mathbf{p} \in P$.

The next theorem links our problem with the auxiliary problem.

THEOREM 5. *(i) If* $\lambda^*$, $\mathbf{p}^* = (p_1^* \ldots p_n^*)^t$ *and* $d^*$ *are optimal to the problem* (1), *then* $d^* = \bar{C}(\lambda^*, \mathbf{p}^*)$ *and* $\lambda^*, \mathbf{p}^*$ *are optimal to* $CTVC(\lambda, \mathbf{p})$.
*(ii) If* $\lambda^*$ *and* $\mathbf{p}^* = (p_1^* \ldots p_n^*)^t$ *are optimal to* $CTVC(\lambda, \mathbf{p})$, *then* $\lambda^*, \mathbf{p}^*$ *and* $d^* = \bar{C}(\lambda^*, \mathbf{p}^*)$ *are optimal to the problem* (1).

*Proof.* (i) From the definition of $\lambda^*$, $\mathbf{p}^*$, $d^*$ and (2) for $\lambda = \lambda^*, d = d^*$, $\mathbf{p} = \mathbf{p}^*$, we obtain $\gamma(\lambda^*, d^*, \mathbf{p}^*) \leqslant \gamma(\lambda^*, \bar{C}(\lambda^*, \mathbf{p}^*), \mathbf{p}^*) = \gamma(\lambda^*, d^*, \mathbf{p}^*) - n(\bar{C}(\lambda^*, \mathbf{p}^*) - d^*)^2$. Therefore, $d^* = \bar{C}(\lambda^*, \mathbf{p}^*)$. Hence, for any $\lambda \in \Pi$ and any $\mathbf{p} \in P$, we have

$$\gamma(\lambda^*, \bar{C}(\lambda^*, \mathbf{p}^*), \mathbf{p}^*) = \gamma(\lambda^*, d^*, \mathbf{p}^*) \leqslant \gamma(\lambda, \bar{C}(\lambda, \mathbf{p}), \mathbf{p}).$$

This means that $\lambda^*$ and $\mathbf{p}^*$ are optimal to $CTVC(\lambda, \mathbf{p})$.

(ii) From the definition of $\lambda^*, \mathbf{p}^*$ and (2), we obtain $\gamma(\lambda^*, \bar{C}(\lambda^*, \mathbf{p}^*), \mathbf{p}^*) \leqslant \gamma(\lambda, \bar{C}(\lambda, \mathbf{p}), \mathbf{p}) = \gamma(\lambda, d, \mathbf{p}) - n(\bar{C}(\lambda, \mathbf{p}) - d)^2 \leqslant \gamma(\lambda, d, \mathbf{p})$, for any $\lambda \in \Pi, \mathbf{p} \in P$ and $d \in D$. This shows that $\lambda^*, \mathbf{p}^*$ and $d^* = \bar{C}(\lambda^*, \mathbf{p}^*)$ are optimal to the problem (1). □

We are now ready to present the algorithm on the basis of Theorems 3 and 5.

### 5.2. DESIGN OF THE PSEUDO-POLYNOMIAL ALGORITHM

Let $V_i$ be the set of all V-shaped sequences with respect to job indices, for jobs $1, 2, \ldots, i$. The following procedure will find an optimal solution for the problem:

$$\min_{\pi \in V_n, 1 \leq \mathbf{p} \leq \mathbf{u}} CTVC(\pi, \mathbf{p}).$$

The idea is that, for each $d \in D$, find $\lambda(d) \in V_n$ and $\mathbf{p}(d) \in P$ such that $\gamma(\lambda(d), d, \mathbf{p}(d)) \leq \gamma(\lambda, d, \mathbf{p})$ for all $\lambda \in V_n$ and $\mathbf{p} \in P$. Then, find the minimum of $\gamma(\lambda(d), d, \mathbf{p}(d))$ for all $d \in D$.

Suppose that a $d \in D$ is given. $\lambda(d)$ and $\mathbf{p}(d)$ are found iteratively using dynamic programming as follows: Consider the partial scheduling for job 1, job 2,..., job $i$. Since $\lambda(d) \in V_n$, job $(i+1), \ldots,$ job $n$ must not be scheduled

among job 1, job 2, ..., job $i$. Suppose that these $i$ jobs start at time $r_i$, under a sequence $\lambda_i \in V_i$, with processing time vector $\mathbf{p}_i = (p_1 \dots p_i)^t \in P_i = \{(p_1 \dots p_i)^t : p_j \in [l_j, u_j], j = 1, \dots, i\}$. Let $C_j(r_i, \lambda_i, \mathbf{p}_i)$ be the completion time of job $j (j = 1, \dots, i)$ under this setting. Define an auxiliary function

$$\gamma_i(\lambda_i, d, r_i, \mathbf{p}_i) = \sum_{j=1}^{i} (C_j(r_i, \lambda_i, \mathbf{p}_i) - d)^2 + \omega \sum_{j=1}^{i} f(u_j - p_j),$$

which is the sum of two parts. The first part is the sum of squared deviations of the completion times of job 1,..., job $i$ from $d$ starting at time $r_i$ under the sequence $\lambda_i \in V_i$. The second part is the compression cost when the processing times are compressed from their upper bounds to $p_j$'s, $j = 1, \dots, i$. It should be clear that $C_j(r_i, \lambda_i, \mathbf{p}_i) = C_j(0, \lambda_i, \mathbf{p}_i) + r_i$. Therefore, $\gamma_i(\lambda_i, d, r_i, \mathbf{p}_i)$ depends only on $\lambda_i, d - r_i$ and $\mathbf{p}_i$. Let $q_i = p_1 + \cdots + p_i \in [L_i, U_i]$, where $L_i = \sum_{j=1}^{i} l_j$ and $U_i = \sum_{j=1}^{i} u_j$. We also let $t_i = d - r_i$. Since $d \in D$ and $r_i$ must be an integer in $[0, U_n - U_i]$, $t_i$ must be an element of

$$T_i = \left\{ -U_n + U_i + 1, -U_n + U_i + 1 + \frac{1}{n}, \dots, U_n - \frac{1}{n}, U_n \right\}.$$

For any given $t_i \in T_i$ and $q_i \in [L_i, U_i]$, we define

$$\beta_i(t_i, q_i) = \min_{\lambda_i \in V_i, \mathbf{p}_i \in Q_i} \gamma_i(\lambda_i, d, r_i, \mathbf{p}_i),$$

where $Q_i = \{(p_1 \dots p_i)^t \in P_i : p_1 + \cdots + p_i = q_i\}$. Since $p_i = q_i - p_1 - \cdots - p_{i-1} \in [q_i - U_{i-1}, q_i - L_{i-1}]$ and $p_i \in [l_i, u_i]$, we have $p_i \in [x, y]$, where $x = \max\{q_i - U_{i-1}, l_i\}$ and $y = \min\{q_i - L_{i-l}, u_i\}$. It is not hard to see that, if job $i$ is scheduled in front of job 1,..., job $(i - 1)$, then

$$\beta_i(t_i, q_i) = \min_{p_i = x, x+1, \dots, y} [\beta_{i-1}(t_i - p_i, q_i - p_i) + (p_i - t_i)^2 + \omega f(u_i - p_i)].$$

On the other hand, if job $i$ is scheduled after job 1,..., job $(i - 1)$, then

$$\beta_i(t_i, q_i) = \min_{p_i = x, x+1, \dots, y} [\beta_{i-1}(t_i, q_i - p_i) + (q_i - t_i)^2 + \omega f(u_i - p_i)].$$

Note that job $i$ cannot be scheduled in any other position because $\lambda(d) \in V_n$. Let

$$\beta 1_i(t_i, q_i) = \min_{p_i = x, x+1, \dots, y} [\beta_{i-1}(t_i - p_i, q_i - p_i) + (p_i - t_i)^2 + \omega f(u_i - p_i)],$$
$$\beta 2_i(t_i, q_i) = \min_{p_i = x, x+1 \dots, y} [\beta_{i-1}(t_i, q_i - p_i) + (q_i - t_i)^2 + \omega f(u_i - p_i)]$$

By the principle of optimality, we have

$$\beta_i(t_i, q_i) = \min\{\beta 1_i(t_i, q_i), \beta 2_i(t_i, q_i)\},$$

for $i = 1, \ldots, n$, subject to $\beta_0(t_0, q_0) = 0, \forall t_0, q_0$.

Note that $r_n = 0, t_n = d$ and $q_n \in [L_n, U_n]$. Therefore,

$$\gamma(\lambda(d), d, \mathbf{p}(d)) = \min_{q_n \in [L_n, U_n]} \beta_n(d, q_n).$$

Since $T_n = D$, we can obtain the optimal $d^*$ by finding the minimum of $\min_{q_n \in [L_n, U_n]} \beta_n(t_n, q_n)$ for $t_n \in T_n$.

It should be noted that the pseudo-polynomial algorithm searches for the best possible $V$-shaped sequence with respect to job indices. Therefore, it will give an optimal solution for any problem instance that (after certain re-labeling of indices) satisfies the agreeable condition.

### 5.3. TIME COMPLEXITY OF THE PSEUDO-POLYNOMIAL ALGORITHM

THEOREM 6. *The time complexity of the pseduo-polynomial algorithm is bounded above by*

$$O(nU(U - L + 1)(U - L + n)).$$

*Proof.* Since $|T_i| = n(2U - U_i - 1) + 1 \leq 2nU, |\{L_i, \ldots, U_i\}| = U_i - L_i + 1 \leq U - L + 1$ and $|\{\max\{q_i - U_{i-l}, l_i\}, \ldots, \min\{q_i - L_{i-1}, u_i\}\}| \leq |\{l_i, \ldots, u_i\}| = u_i - l_i + 1$, it can easily be seen that the time complexity of the algorithm is bounded above by

$$O\left(\sum_{i=1}^{n} (2nU)(U - L + 1)(u_i - l_i + 1)\right) = O(nU(U - L + 1)(U - L + n)). \quad \square$$

COROLLARY 1. *If there exists a constant $c$ such that $u_i - l_i \leq c$ for all $i = 1, \ldots, n$, then the time complexity is bounded above by $O(n^3 U)$.*

*Proof.* It follows by noting that $U - L \leq nc$. $\square$

Corollary 1 indicates that if the range of each processing time is bounded by a constant, then the time complexity reduces to $O(n^3 U)$.

COROLLARY 2. *If there exists a constant $C$ such that $U - L \leq C$, then the time complexity is bounded above by $O(n^2 U)$.*

*Proof.* It follows directly from Theorem 6. $\square$

Corollary 2 indicates that if the total variation of the processing times is bounded, then the time complexity reduces to $O(n^2 U)$. The classical CTV problem is a special case of this situation, where $U - L = 0$.

## 6. Heuristic Algorithms for CTVC Problem

In this section, we aim at devising heuristic algorithms for the general problem. In the first heuristic algorithm, we construct modified lower bounds to make the problem agreeable.

*Heuristic 1 (H1)*:

1. Sort the jobs so that the upper bounds are in non-decreasing order, i.e., $u_1 \leq \cdots \leq u_n$.
2. Set $\tilde{l_1} = l_1, \tilde{l_i} = \max\{l_i, \tilde{l_{i-l}}\}, i = 2, \ldots, n$.
3. Use the pseduo-polynomial algorithm in Section 5 for the modified lower bounds to find a solution.

It is clear that $\tilde{l_1} \leq \cdots \leq \tilde{l_n}$. Therefore, the problem with the modified lower bounds satisfies the agreeable condition. Hence, Step 3 will give an optimal solution for the modified problem. But the solution may not be optimal for the original problem. We will provide an error analysis for H1 in the following. The next lemma will be used in the analysis.

LEMMA 2. *Let* $B = (b_{ij})$ *be an* $m \times m$ *square matrix, and* $\mathbf{x} = (x_1 \ldots x_m)^t$, $\mathbf{y} = (y_1 \cdots y_m)^t$ *and* $\mathbf{z} = (z_1 \ldots z_m)^t$ *be three* $m \times 1$ *column vectors, where all* $b_{ij}, x_i, y_i$ and $z_i$ *are positive real numbers. Then,*

$$\frac{\mathbf{x}^t B \mathbf{y}}{\mathbf{z}^t B \mathbf{z}} \leq \left( \max_{i=1,\ldots,m} \left\{ \frac{x_i}{z_i} \right\} \right) \left( \max_{i=1,\ldots,m} \left\{ \frac{y_i}{z_i} \right\} \right).$$

*Proof.* Let $k = \max_{i=1,\ldots,m} \left\{ \frac{x_i}{z_i} \right\}$ and $l = \max_{i=1,\ldots,m} \left\{ \frac{y_i}{z_i} \right\}$. Then, $x_i \leq k z_i$ and $y_i \leq l z_i$, for $i = 1, \ldots, m$. Thus,

$$\frac{\mathbf{x}^t B \mathbf{y}}{\mathbf{z}^t B \mathbf{z}} = \frac{\sum_{i,j=1,\ldots,m} b_{ij} x_i y_j}{\sum_{i,j=1,\ldots,m} b_{ij} z_i z_j} \leq \frac{kl \sum_{i,j=1,\ldots,m} b_{ij} z_i z_j}{\sum_{i,j=1,\ldots,m} b_{ij} z_i z_j} = kl. \qquad \square$$

We will analyze the relative error and obtain a bound for it. After Step 1 of $H1$, the $n$ jobs have been sorted so that $u_1 \leq \ldots \leq u_n$. Let $\mathbf{l} = (l_1 \ldots l_n)^t$ and $\mathbf{u} = (u_1 \cdots u_n)^t$. Note that the CTV part involves only square terms. Therefore, it can be expressed as a quadratic form involving all $p_i$ and the sequence. Let $A = (a_{ij})$ be the $(n-1) \times (n-1)$ symmetric matrix defined in [9]. Accordingly,

$$a_{ij} = \begin{cases} i(n-j), & i \leqslant j, \\ j(n-i), & i > j. \end{cases}$$

Let $\mathbf{p} = (p_1 \cdots p_n)^t$ and, for any sequence $\pi$, let $\mathbf{p}_\pi = \left(p_{\pi(2)} \cdots p_{\pi(n)}\right)^t$. By [9], the CTV part can be expressed as $\frac{1}{n} \mathbf{p}_\pi^t A \mathbf{p}_\pi$. Therefore, the original problem can be expressed as minimizing

$$CTVC(\pi, \mathbf{p}) = \frac{1}{n} \mathbf{p}_\pi^t A \mathbf{p}_\pi + \omega \sum_{i=1}^{n} f(u_i - p_i),$$

subject to $\pi \in \Pi$ and $\mathbf{l} \le \mathbf{p} \le \mathbf{u}$. We should note that $p_{\pi(1)}$ is not involved in the quadratic form, which implies that for any optimal sequence $\pi^*$ we can have (i) $p_{\pi^*(1)} \ge p_{\pi^*(i)}, i = 2, \ldots, n$, and (ii) $p_{\pi^*(1)} = u_{\pi^*(1)}$. Let $\tilde{\mathbf{l}} = (\tilde{l_1} \ldots \tilde{l_n})^t$. Suppose that

$$CTVC(\pi^\circ, \mathbf{p}^\circ) = \min_{\pi \in \Pi, \tilde{\mathbf{l}} \le \mathbf{p} \le \mathbf{u}} CTVC(\pi, \mathbf{p})$$

and

$$CTVC(\pi^*, \mathbf{p}^*) = \min_{\pi \in \Pi, \mathbf{l} \le \mathbf{p} \le \mathbf{u}} CTVC(\pi, \mathbf{p}).$$

Since $\mathbf{l} \le \tilde{\mathbf{l}}$, we have $CTVC(\pi^\circ, \mathbf{p}^\circ) \ge CTVC(\pi^*, \mathbf{p}^*)$. The relative error of H1 is therefore

$$e = \frac{CTVC(\pi^\circ, \mathbf{p}^\circ) - CTVC(\pi^*, \mathbf{p}^*)}{CTVC(\pi^*, \mathbf{p}^*)}$$

Let $\delta_i = \tilde{l_i} - l_i, i = 1, \ldots, n$ and $r = \max\left\{\frac{\delta_1}{l_1}, \ldots, \frac{\delta_n}{l_n}\right\}$. Also, let $\delta = 2r + r^2$. We have the following result.

**LEMMA 3.**

$$e \le \delta.$$

*Proof.* Let $\Delta_i = \max\{0, \tilde{l_i} - p_i^*\}, i = 1, \ldots, n$, and $\Delta = (\Delta_1 \ldots \Delta_n)^t$. We define $\tilde{\mathbf{p}} = (\tilde{p_1} \ldots \tilde{p_n})^t = \mathbf{p}^* + \Delta$. Note that $0 \le \Delta_i \le \delta_i, i = 1, \ldots, n$, and $\tilde{\mathbf{l}} \le \tilde{\mathbf{p}} \le \mathbf{u}$. Using the non-decreasing property of $f$, we have $f(u_i - \tilde{p_i} + \Delta_i) \ge f(u_i - \tilde{p_i}), i = 1, \ldots, n$. Therefore,

$$\begin{aligned}
CTVC(\pi^*, \mathbf{p}^*) &= \frac{1}{n}(\mathbf{p}^*)^t_{\pi^*} A(\mathbf{p}^*)_{\pi^*} + \omega \sum_{i=1}^{n} f(u_i - p_i^*) \\
&= \frac{1}{n}(\tilde{\mathbf{p}} - \Delta)^t_{\pi^*} A(\tilde{\mathbf{p}} - \Delta)_{\pi^*} + \omega \sum_{i=1}^{n} f(u_i - \tilde{p_i} + \Delta_i) \\
&\ge \frac{1}{n}(\tilde{\mathbf{p}} - \Delta)^t_{\pi^*} A(\tilde{\mathbf{p}} - \Delta)_{\pi^*} + \omega \sum_{i=1}^{n} f(u_i - \tilde{p_i}) \\
&= CTVC(\pi^*, \tilde{\mathbf{p}}) - h(\pi^*, \mathbf{p}^*) \\
&\ge CTVC(\pi^\circ, \mathbf{p}^\circ) - h(\pi^*, \mathbf{p}^*),
\end{aligned}$$

where $h(\pi^*, \mathbf{p}^*) = \frac{2}{n}\Delta^t_{\pi^*} A(\tilde{\mathbf{p}})_{\pi^*} - \frac{1}{n}\Delta^t_{\pi^*} A \Delta_{\pi^*} = \frac{2}{n}\Delta^t_{\pi^*} A \mathbf{p}^*_{\pi^*} + \frac{1}{n}\Delta^t_{\pi^*} A \Delta_{\pi^*}$. We have,

1. $CTVC(\pi^\circ, \mathbf{p}^\circ) - CTVC(\pi^*, \mathbf{p}^*) \le h(\pi^*, \mathbf{p}^*) = \frac{2}{n}\Delta^t_{\pi^*} A \mathbf{p}^*_{\pi^*} + \frac{1}{n}\Delta^t_{\pi^*} A \Delta_{\pi^*}$, and
2. $CTVC(\pi^*, \mathbf{p}^*) = \frac{1}{n}(\mathbf{p}^*)^t_{\pi^*} A(\mathbf{p}^*)_{\pi^*} + \omega \sum_{i=1}^{n} f(u_i - p_i^*) \ge \frac{1}{n}(\mathbf{p}^*)^t_{\pi^*} A(\mathbf{p}^*)_{\pi^*} > 0.$

Hence, by Lemma 2,

$$
\begin{aligned}
e &= \frac{CTVC(\pi^\circ, \mathbf{p}^\circ) - CTVC(\pi^*, \mathbf{p}^*)}{CTVC(\pi^*, \mathbf{p}^*)} \\
&\le \frac{2\Delta_{\pi^*}^t A \mathbf{p}_{\pi^*}^*}{(\mathbf{p}^*)_{\pi^*}^t A(\mathbf{p}^*)_{\pi^*}} + \frac{\Delta_{\pi^*}^t A \Delta_{\pi^*}}{(\mathbf{p}^*)_{\pi^*}^t A(\mathbf{p}^*)_{\pi^*}} \\
&\le 2 \max_i \left(\frac{\Delta_i}{p_i^*}\right) + \left\{\max_i \left(\frac{\Delta_i}{p_i^*}\right)\right\}^2 \\
&\le 2 \max_i \left(\frac{\Delta_i}{l_i}\right) + \left\{\max_i \left(\frac{\Delta_i}{l_i}\right)\right\}^2 \\
&= \delta
\end{aligned}
$$

□

In the above analysis, we investigate the relative error when the pseudo-polynomial algorithm of Section 5 is applied for the modified lower bounds. However, we can apply the pseudo-polynomial algorithm for the original lower bounds as follows. Let $\Omega$ be the set of all V-shaped sequences with respect to the job indices, and $CTVC(\pi', \mathbf{p}') = \min_{\pi \in \Omega, \mathbf{l} \le \mathbf{p} \le \mathbf{u}} CTVC(\pi, \mathbf{p})$. We have,

$$
\begin{aligned}
CTVC(\pi', \mathbf{p}') &= \min_{\pi \in \Omega, \mathbf{l} \le \mathbf{p} \le \mathbf{u}} CTVC(\pi, \mathbf{p}) \\
&\le \min_{\pi \in \Omega, \tilde{\mathbf{l}} \le \mathbf{p} \le \mathbf{u}} CTVC(\pi, \mathbf{p}) \\
&= \min_{\pi \in \Pi, \tilde{\mathbf{l}} \le \mathbf{p} \le \mathbf{u}} CTVC(\pi, \mathbf{p}) = CTVC(\pi^\circ, \mathbf{p}^\circ)
\end{aligned}
$$

Let $e1$ be the relative error when we apply the pseudo-polynomial algorithm for the orginal lower bounds, but only considering those sequences in $\Omega$. We get,

$$
e1 = \frac{CTVC(\pi', \mathbf{p}') - CTVC(\pi^*, \mathbf{p}^*)}{CTVC(\pi^*, \mathbf{p}^*)} \le \frac{CTVC(\pi^\circ, \mathbf{p}^\circ) - CTVC(\pi^*, \mathbf{p}^*)}{CTVC(\pi^*, \mathbf{p}^*)} = e \le \delta.
$$

Once we have obtained the solution $(\pi', \mathbf{p}')$, we can take an extra step to improve it. We can treat $\mathbf{p}'$ as a given vector of processing times, and then find the best sequences $\pi''$ to minimize $CTVC(\pi, \mathbf{p}')$ for all $\pi \in \Pi$, i.e.,

$$
CTVC(\pi'', \mathbf{p}') = \min_{\pi \in \Pi} CTVC(\pi, \mathbf{p}').
$$

Therefore

$$
\begin{aligned}
CTC(\pi'', \mathbf{p}') + CC(\mathbf{p}') &= \min_{\pi \in \Pi} [CTV(\pi, \mathbf{p}') + CC(\mathbf{p}')] \\
&= \left[\min_{\pi \in \Pi} CTV(\pi, \mathbf{p}')\right] + CC(\mathbf{p}').
\end{aligned}
$$

Hence,

$$CTV(\pi'', \mathbf{p}') = \min_{\pi \in \Pi} CTV(\pi, \mathbf{p}').$$

Thus, to find $\pi''$ is just a classical CTV minimization problem when $\mathbf{p}'$ is considered as a given vector of processing times.

In the above discussion, we first sort the jobs so that the upper bounds are in non-decreasing order, i.e., $u_1 \leq \cdots \leq u_n$. It seems a good idea to first sort the jobs so that the lower bounds are in non-decreasing order, i.e., $l_j \leq \cdots \leq l_n$, and then apply the steps in the above discussion. This suggests a heuristic algorithm that does the two ways of calculation and then takes the minimum between the two results. To summarize, the heuristic algorithm is as follows.

*Heuristic 2 (H2)*:

1. Sort the jobs so that the upper bounds are in non-decreasing order.
2. Use the pseudo-polynomial algorithm in Section 5 to find a solution.
3. Given the processing time vector obtained in Step 2, find an optimal sequence w.r.t. this processing time vector by solving a CTV problem.
4. Record the result.
5. Sort the jobs so that the lower bounds are in non-decreasing order.
6. Repeat Steps 2–4.
7. Compare the two results and retain the better one.

Obviously, the time complexity of H2 is the same as that of H1, i.e., $O(nU(U-L+1)(U-L+n))$. But, the relative error $R$ of H2 is less than or equal to that of H1, i.e., $e$. Therefore, R is also bounded above by $\delta$, and H2 is chosen for solving the CTVC problem in general.

## 7. Computational Results

In the previous section, we have shown that $R \leq \delta$. However, $\delta = 2r + r^2$ is only a worst-case upper bound for $R$. Practically, the relative error may be much smaller than this. In this section, the practical performance of the heuristic algorithm H2 is evaluated by implementing this algorithm to solve a number of testing problems.

The algorithm was coded in $C$ and experiments were conducted to evaluate the performance of the algorithm on a Linux-based PC with P4 2.4 GHz CPU and 1 GB physical memory. Problems of 5–80 jobs were randomly generated. The bounds of the processing times were randomly generated from [1,15]. The weights $\omega$ are in the set $\{0.2, 0.4, 0.6, 0.8, 1, 5, 10\}$. A total of five instances were randomly generated for each problem size. For each problem instance, the heuristic algorithm H2 was applied to derive an approximated solution. Since the problem is NP-hard, the

optimal solutions for large problem sizes are unlikely obtained in reasonable time, so the quality of the approximated solution was evaluated in the following ways:

- For problems with $n \leq 9$, the solutions obtained by the heuristic algorithm $H2$ were compared with the solutions obtained by complete enumeration and the compression cost function was taken to be $f(x) = x^2$.
- For problems with $n > 9$, the solutions obtained by the heuristic algorithm H2 were compared with the tight lower bound obtained by the heuristic algorithm H1, and the compression cost function was a convex non-decreasing function:

$$f(x) = \begin{cases} 8x, & 0 \leq x \leq 4 \\ 2x^2, & x > 4 \end{cases}.$$

(Note: To show that the algorithm is applicable to any convex non-decreasing function, two different cost functions were used in the experiments.)

The results of the experiments are reported in the following. First, the computational results for $n \leq 9$ are summarized in Table 1, in which $\bar{R}$ is the average relative error of the five solutions found by H2 for each $n$. Also included in the table are the average CPU time required to solve the problems using the heuristic algorithm H2 and complete enumeration. The results show that the heuristic algorithm H2 gives near optimal solutions when the weights $\omega$ are small, although there is a trend that the average relative error increases with the weight. On average, the algorithm took less than 0.3 s to find the solution for $n \leq 9$, which is much more efficient than complete enumeration.

Second, for bigger problems (i.e., $n > 9$), it becomes infeasible to find the optimal solution by complete enumeration for evaluating the performance of the heuristic algorithm H2. Instead, the tight lower bound obtained by the heuristic algorithm H1 can be used to derive the relative

*Table 1.* Performance of the heuristic algorithm H2 for $n \in [2, 9]$

| $n$ | $\bar{R}$ for $\omega$ equal to | | | | | | | Average CPU time (s) | |
|---|---|---|---|---|---|---|---|---|---|
| | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 5.0 | 10.0 | H2 | Enumeration |
| 2 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.0 | 0.0 |
| 3 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.0 | 0.0 |
| 4 | 0.000 | 0.000 | 0.000 | 0.032 | 0.000 | 0.000 | 0.000 | 0.0 | 0.1 |
| 5 | 0.000 | 0.000 | 0.000 | 0.001 | 0.003 | 0.001 | 0.006 | 0.0 | 0.7 |
| 6 | 0.000 | 0.000 | 0.002 | 0.000 | 0.027 | 0.008 | 0.009 | 0.1 | 11.2 |
| 7 | 0.000 | 0.000 | 0.000 | 0.011 | 0.011 | 0.003 | 0.018 | 0.1 | 69.2 |
| 8 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.006 | 0.022 | 0.2 | 342.3 |
| 9 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.013 | 0.021 | 0.3 | 470.6 |

error bound of the heuristic algorithm H2. Since the heuristic algorithm H2 is a dynamic programming algorithm, the program must keep the values of all state variables in order to back-track the processing times and the sequence of the solution. Given an $O(nU(U-L+1)(U-L+n))$ complexity, the computer memory for maintaining the essential data for back-tracking would be quite taxing. With 1GB physical memory, the PC could easily handle problems up to 50 jobs. The results for problems ranging from 10 to 50 jobs are given in Table 2, in which $\bar{B}$ is the average relative error bound of the five solutions found by the heuristic algorithm H2 for each $n$ tested. The average CPU time to find the solution for each $n$ is also given in the table. Similar to the results for $n \leq 9$, the relative error bound increases with the weight. Moreover, the relative error bound is oscillating diminishingly as $n$ grows, and the algorithm is reasonably efficient as it can find the solution for $n = 50$ in 378.6 s on average.

Third, to find the processing times and the sequence of the solution for problems of more than 50 jobs, the PC must have more than 1GB physical memory. However, calculating the relative error bound only needs the objective values but not the processing times and the sequence of the solution. So, back-tracking could be avoided if we do not apply the CTV algorithm to enhance the solution obtained by the pseudo-polynomial algorithm (i.e., Step 3 of the heuristic algorithm H2). Although bypassing Step 3 may affect the solution quality, it greatly reduces the computer memory required for keeping the state information for computing the objective values of the solution. Based in this simplification, we successfully evaluated the performance of the heuristic algorithm H2 for problems from 60 to 80 jobs using the PC. The results obtained are tabulated in Table 3. From the table, one can see that bypassing Step 3 only reduces the average time slightly and it does not significantly affect the performance of the heuristic algorithm H2 when $n$ is large. In addition, the trends of the relative error bound with respect to the weights and $n$ persist and closely follow

*Table 2.* Perfomance of the heuristic algorithm H2 for $n \in [10, 50]$

| $n$ | $\bar{R}$ for $\omega$ equal to | | | | | | | Average CPU time (s) |
|---|---|---|---|---|---|---|---|---|
| | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 5.0 | 10.0 | |
| 10 | 0.040 | 0.046 | 0.099 | 0.043 | 0.109 | 0.103 | 0.045 | 0.5 |
| 15 | 0.011 | 0.035 | 0.029 | 0.038 | 0.035 | 0.183 | 0.130 | 2.8 |
| 20 | 0.007 | 0.013 | 0.013 | 0.019 | 0.039 | 0.111 | 0.139 | 8.2 |
| 25 | 0.004 | 0.012 | 0.013 | 0.022 | 0.024 | 0.084 | 0.135 | 21.2 |
| 30 | 0.005 | 0.005 | 0.009 | 0.011 | 0.017 | 0.067 | 0.102 | 43.7 |
| 35 | 0.003 | 0.004 | 0.009 | 0.009 | 0.013 | 0.046 | 0.082 | 89.0 |
| 40 | 0.002 | 0.003 | 0.006 | 0.007 | 0.011 | 0.043 | 0.091 | 157.2 |
| 45 | 0.002 | 0.003 | 0.005 | 0.006 | 0.007 | 0.030 | 0.083 | 233.3 |
| 50 | 0.001 | 0.002 | 0.004 | 0.005 | 0.005 | 0.034 | 0.061 | 378.6 |

*Table 3.* Perfomance of the heuristic algorithm H2 (without solving the CTV problem) for $n \in [60, 80]$

| $n$ | $\bar{B}$ for $\omega$ equal to | | | | | | | Average CPU time (s) |
|---|---|---|---|---|---|---|---|---|
| | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 5.0 | 10.0 | |
| 60 | 0.001 | 0.002 | 0.002 | 0.003 | 0.004 | 0.023 | 0.039 | 322.4 |
| 70 | 0.001 | 0.001 | 0.002 | 0.003 | 0.003 | 0.015 | 0.031 | 1299.2 |
| 80 | 0.001 | 0.001 | 0.002 | 0.002 | 0.002 | 0.015 | 0.025 | 1877.9 |

those in Table 2. To help visualize the trend, the average relative error bounds of H2 for problems from 10 to 80 jobs are plotted in Figure 1. The figure clearly shows that the heuristic algorithm can give near optimal solutions when $n$ is large.

## 8. Concluding Remarks

We have studied a new CTV model in which the processing times of the jobs can be compressed within certain intervals and the compression cost is formulated as a general convex non-decreasing function of the amount of processing time compressed. A pseudo-polynomial algorithm of time complexity bounded above by $O(nU(U-L+1)(U-L+n))$ is derived to solve the problem under an agreeable condition. Basing on the pseudo-polynomial algorithm, we obtain two heuristic algorithm H1 and H2 for solving the general problem. Although H1 can find an optimal solution for the agreeable

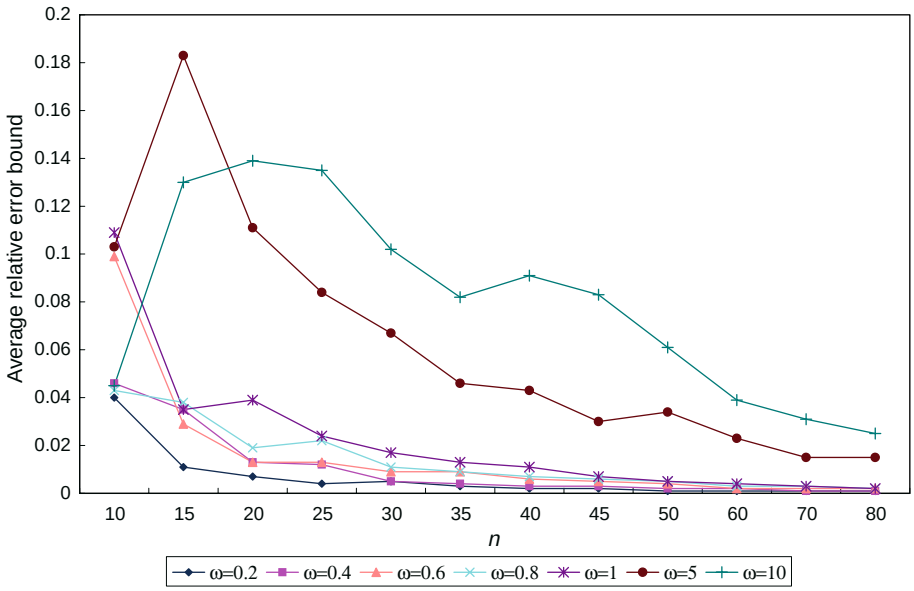


*Figure 1.* Trend of average relative error bounds of the heuristic algorithm H2.

problem, H2 is dominant for solving the general problem. The performance of H2 has been evaluated by applying it to solve a number of randomly generated problem instances. The tight lower bound derived in Section 4 has been used for evaluating the performance. The results show that H2 in general can efficiently yield near optimal solutions, when $n$ is large.

Topics for future research include: (a) relaxing the agreeable condition; (b) proving that the problem is NP-hard in the strong sense or in the weak sense; (c) generalizing the results to multi-machine problems; and (d) generalizing the results to other earliness/tardiness scheduling problems.

## 9. Acknowledgements

This research was supported in part by The Hong Kong Polytechnic University under grant number G-T596.

## References

1. Alidaee, B. and Ahmadian, A. (1993), Two Parallel Machine Sequencing Problems Involving Controlled Job Processing Times, *European Journal of Operational Research*, 70, 335–341.
2. Cai, X. (1995), Minimization of agreeably weighted variance in single machine systems, *European Journal of Operational Research* 85, 576–592.
3. Cheng, T.C.E., Chen, Z.L. and Li, C.L. (1996), Parallel-machine scheduling with controllable processing times, *IIE Transactions* 28,177–180.
4. Cheng, T.C.E., Chen, Z.L. and Li, C.L. (1996), Single-machine scheduling with tradeoff between number of trady jobs and resource allocation, *Operation Research Letters* 19, 237–242.
5. De, P., Ghosh, J.B. and Wells, C.E. (1992), On the minimization of completion time variance with a bicriteria extension, *Operation Research* 40(6), 1148–1155.
6. Eilon, S. and Chowdhury, I.G. (1977), Minimizing waiting time variance in the single machine problem, *Management Science* 23, 567–575.
7. Kubiak, W. (1993), Completion time variance minimization on a single machine is difficult, *Operation Research Letters* 14, 49–59.
8. Merten, A.G. and Muller, M.E. (1972), Variance minimization in single machine sequencing problems, *Management Science* 18, 518–528.
9. Ng, C.T., Cai, X. and Cheng, T.C.E. (1999), Probabilistic analysis of an asymptotically optimal solution for the completion time variance problem, *Naval Research Logistics* 46, 373–398.
10. Ng, C.T., Cai, X. and Cheng, T.C.E. (1996), A tight lower bound for the completion time variance problem, *European Journal of Operational Research* 92, 211–213.
11. Nowicki, E. and Zdrzalka, S. (1980), A Two-machine flow shop scheduling problems with controllable job processing times, *European Journal of Operational Research* 34, 208–220.
12. Nowicki, E. and Zdrzalka, S. (1990), A survey of results for sequencing problems with controllable processing times, *Discrete Applied Mathematics* 26, 271–287.
13. Panwalkar, S.S. and Rajagopalan, R. (1992), Single-machine sequencing with controllable processing times, *European Journal of Operational Research* 59, 298–302.