

SINGLE MACHINE SCHEDULING WITH CONTROLLABLE PROCESSING TIMES AND COMPRESSION COSTS (PART I: EQUAL TIMES AND COSTS)

TANG GUOCHUN AND FOULDS, L. R.

Abstract. Most papers in scheduling research have treated individual job processing times as fixed parameters. However, in many practical situations, a manager may control processing time by reallocating resources. In this paper, authors consider a machine scheduling problem with controllable processing times. In the first part of this paper, a special case where the processing times and compression costs are uniform among jobs is discussed. Theoretical results are derived that aid in developing an $O(n^2)$ algorithm to solve the problem optimally. In the second part of this paper, authors generalize the discussion to general case. An effective heuristic to the general problem will be presented.

§ 1 Introduction

This paper considers the problem of simultaneous determination of optimal compression activities and optimal sequence for N jobs to be processed on a single machine. All jobs are available at time zero. Processing times are known and deterministic. However, we assume that the processing times of jobs can be crashed for uniform amounts of time and cost. The objective is to simultaneously determine an optimal sequence and the optimal crashing activities to minimize the total costs in completion and compression.

Early in 1969, Lawler and Moore^[1] studied the single machine scheduling problem with two different machine processing times and costs for each job. This single machine problem with controllable machine processing time and cost has been further studied by Vickson^[2,3], Van Wassenhove and Baker^[4], Tuzikov^[5], Ruiz Diaz and French^[6], etc. Readers are directed to Nowicki and Zdrzalka^[7] for a detailed review of the literature relat-

Received September 1, 1997.

1991 *MR Subject Classification*: 68Q25.

Keywords: Machine scheduling problems, controllable processing times, uniform compression time and cost, dominance set, lateness, crash activities, polynomial time algorithm.

Supported by the National Natural Science Foundation of China (69484003) to the first author.

ed to this kind of problems. The problem considered in this paper is most directly related to the problems considered in [2]. Considered a single machine problem with controllable processing time. The objective of the problem was to minimize total weighted completion times. He showed that in the case that each job can either be processed in a normal time at zero cost or in maximally compressed time at a positive compression cost. The complexity of this problem is still unknown. He assumed that the compression costs are linear functions of processing times of jobs. To solve the problem, he first developed the analytical conditions to identify job status (to be crashed or not to be crashed) and if the job status can not be identified by the conditions, a heuristic was used to find an approximate optimal solution.

In the first part of our paper, we consider a special case in which the allowable compression time and related compression costs are uniform among jobs. To our knowledge, no paper has been found in examining this topic. We provide $O(n^2)$ algorithm to solve the problem optimally. By this, we mean that the problem, in this case, is polynomially solvable. Both theoretical and numerical results are presented.

We will generalize our discussion and the algorithm to general case in the second part of this paper, where we will present some computational numerical results.

Part I is organized as follows. Section 2 provides some basic analysis. In Section 3, we propose some important theoretical results that lead to the development of an algorithm that is presented in Section 4. A numerical example to illustrate the algorithm is described in Section 5. Conclusions and extensions are discussed in Section 6. We omit some lengthy proofs of some theorems and lemmas. For the detail proof readers can refer to [8].

§ 2 Problem Formulation and Basic Results

We assume that there are n jobs to be processed on a single machine one at a time. Each job has its normal processing time which can be compressed. For each job, a positive weight is assigned to represent its importance. There are therefore two types of costs:

1. the completion (including waiting) time costs,
2. the compression costs.

The objective of our problem is to find an optimal sequence and an optimal decision of compression activities to minimize the sum of 1 and 2.

We now introduce some notation used in the rest of this paper. Let $J = \{1, 2, \dots, n\}$ be the job set. For each job $j \in J$, let

p_j = the normal processing time, $p_j > 0$, u_j = the maximum compression time, $0 \leq u_j \leq p_j$;
 x_j = the actual amount of time by which job j is compressed, with $0 \leq x_j \leq u_j$, the vector of the compression activities of all jobs is denoted by $x = (x_1, x_2, \dots, x_n)$, and the set of all feasible compression activities is denoted by $X = \{x = (x_1, x_2, \dots, x_n) : 0 \leq x_j \leq u_j, j$

$\in J$ }, and then t_j =the actual processing time, with $t_j = p_j - x_j$;
 c_j =the compression cost per unit time, w_j =the completion cost (weight) per unit time;
 s =a permutation (sequence) of J , i. e. the i th job to be processed in sequence s is denoted as $s(i)$, the set of all permutations of J is denoted as S ;
 C_j =the completion time, for given compression vector x and sequence s ,

$$C_{s(j)} = \sum_{i=1}^j t_{s(i)} = \sum_{i=1}^j (p_{s(i)} - x_{s(i)}).$$

For a given compression vector x , the total compression cost is $F_2(x) = \sum_{j=1}^n c_j x_j$.

Note that x_j (for all j) is independent of the sequence s . For each decision pair (x, s) , the total completion cost can be described as below:

$$F_1(x, s) = \sum_{j=1}^n w_{s(j)} C_{s(j)} = \sum_{j=1}^n w_{s(j)} \sum_{i=1}^j t_{s(i)} = \sum_{j=1}^n w_{s(j)} \sum_{i=1}^j (p_{s(i)} - x_{s(i)}).$$

Therefore, the total cost is $F(x, s) = F_1(x, s) + F_2(x)$. Our problem is to find an optimal $(x^*, s^*) \in (X, S)$ which minimizes F , i. e.

$$F(x^*, s^*) = \min_{(x, s) \in (X, S)} F(x, s)$$

The complexity of this problem is still unknown, although many authors believe that it is NP-hard. (see, for example, [2] and [7].) We now discuss a special case in which the maximum compression time u_j and unit compression cost c_j are equal for all jobs. That is $c_j = c, u_j = u$, for all $j = 1, 2, \dots, n$.

For any fixed sequence s , it has been shown that^[2] the following x minimizes $F(x, s)$:

$$x_{s(j)} = \begin{cases} u, & \text{when } c < \sum_{i=j}^n w_{s(i)}, \\ 0, & \text{when } c \geq \sum_{i=j}^n w_{s(i)}. \end{cases} \quad (1)$$

This result tells us that for each job we only need to consider two possibilities; being crashed ($x_j = u$) or being uncrashed ($x_j = 0$). In (1), we also see that if $c < \sum_{i=j}^n w_{s(i)}$ for the j th job in sequence s , which implies $x_{s(j)} = u$, then $c < \sum_{i=k}^n w_{s(i)}$ for all $k < j$ and hence $x_{s(k)} = u$ for all jobs before the j th job in the sequence s . This means that all crashed jobs should be put before uncrashed jobs in an optimal solution.

On the other hand, for any fixed compression (or crash) vector x , the optimal sequence that minimizes $F_1(x, s)$ is the corresponding weighted shortest processing time (WSPT)^[9] sequence. This implies that only x is an independent variable. When x changes, the related WSPT sequence changes correspondingly. Since both x and s are decision variables in the original problem, we still keep two variables x and s in the objective function $F(x, s)$, as well as in the decision pair (x, s) , although only x is independent.

Our proposed $O(n^2)$ algorithm is a procedure to decide x . When x is determined, the corresponding WSPT sequence can be found in $O(n \log n)$ time.

We define a feasible solution (x, s) as below, satisfying:

- 1) $x_j = 0$ or u , for all $j = 1, 2, \dots, n$;

2) s is in WSPT order with respect to the actual processing times $\{t_j = p_j - x_j, j = 1, 2, \dots, n\}$;

3) all crashed jobs are put before uncrashed jobs.

Obviously, an optimal solution can be found among feasible solutions.

In the rest of the paper we will use the following notation.

- Denote $\{\underline{1}, \underline{2}, \dots, \underline{n}\}$ as the corresponding crashed jobs.

- Write $i \leftarrow j$ if $\frac{p_i}{w_i} \leq \frac{p_j}{w_j}$ or $p_i \leq p_j$ when $\frac{p_i}{w_i} = \frac{p_j}{w_j}$. We assume $\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \dots \leq \frac{p_n}{w_n}$, or $p_j \leq p_{j+1}$ when $\frac{p_j}{w_j} = \frac{p_{j+1}}{w_{j+1}}$. That is $1 \leftarrow 2 \leftarrow \dots \leftarrow n$. We also assume that for any feasible solution (x, s) , $\frac{t_{s(j)}}{w_{s(j)}} = \frac{t_{s(j+1)}}{w_{s(j+1)}}$ implies $s(j) < s(j+1)$.

- Use $j \downarrow$ to denote the decision to crash job j when it is originally uncrashed (i. e. change $x_j = 0$ to $x_j = u$), and $j \uparrow$ as the decision to resume (or recover) job j when it is originally crashed (i. e. change $x_j = u$ to $x_j = 0$).

- For jobs i and j , if $i \leftarrow j$, we have 3 cases when both i and j are crashed:

- We write $i \leftarrow\leftarrow j$, if $\frac{p_i - u}{w_i} \leq \frac{p_j - u}{w_j} < \frac{p_i}{w_i} \leq \frac{p_j}{w_j}$. Clearly, this means that $\underline{i} \leftarrow\leftarrow \underline{j} \leftarrow\leftarrow i \leftarrow\leftarrow j$.

- We write $i \leftarrow\leftarrow\leftarrow j$, if $\frac{p_i - u}{w_i} < \frac{p_i}{w_i} \leq \frac{p_j - u}{w_j} < \frac{p_j}{w_j}$. This means that $\underline{i} \leftarrow\leftarrow\leftarrow \underline{i} \leftarrow\leftarrow\leftarrow \underline{j} \leftarrow\leftarrow\leftarrow j$.

- We write $i \leftarrow j$, if $\frac{p_j - u}{w_j} < \frac{p_i - u}{w_i} < \frac{p_i}{w_i} \leq \frac{p_j}{w_j}$. This means that $\underline{j} \leftarrow\leftarrow \underline{i} \leftarrow\leftarrow i \leftarrow j$. In this case, we know $w_i > w_j$, and $p_i > p_j$.

- For a given solution (x, s) , with s in WSPT sequence, if we change the status of the job k and the WSPT sequence changes accordingly, we use (x', s') to denote the new solution. We use the notation $E_k(x, s)$ for the increment (can be negative) of the cost function when we change the status of job k , i. e. $E_k(x, s) = F(x', s') - F(x, s)$. For any given (x, s) , there are at most n possible increments $E_k(x, s), k = 1, 2, \dots, n$.

- For any solution (x, s) , we denote $L(x, s)$ as the set of crashed jobs, i. e. $L(x, s) = \{j \in J | x_j = u\}$.

When all jobs are uncrashed, i. e. $x = (0, 0, \dots, 0)$, we assume $s^0 = (1, 2, \dots, n)$ is the initial sequence. We now adopt Vickson's notation and let $E_j(0, s^0) = D_j$. Also by Vickson's result we have

$$E_j(0, s^0) = D_j = u(c - w_j - \sum_{k \in \alpha_j} w_k) + p_j \sum_{k \in \beta_j} w_k - w_j \sum_{k \in \beta_j} p_k, \tag{2}$$

where α_j is the set of jobs following j after j is crashed and repositioned, while β_j is the possibly empty set of jobs lying between the old and new positions of job j . i. e.

$$\alpha_j = \left\{ k \neq j \mid \frac{p_k}{w_k} \geq \frac{p_j - u}{w_j} \right\}, \text{ and } \beta_j = \left\{ k \neq j \mid \frac{p_j - u}{w_j} \leq \frac{p_k}{w_k} \leq \frac{p_j}{w_j} \right\}.$$

When i and $j(i \neq j)$ are both uncrashed, we use α_{ij} to denote the increment of D , when i is crashed. Vickson showed that

$$\alpha_{ij} = \begin{cases} w_j p_i - w_i(p_j - u) > 0, & \text{when } i \leftarrow\leftarrow j, \\ 0, & \text{when } i \leftarrow j, \\ w_j u > 0, & \text{when } i \leftarrow j, \\ w_i p_j - w_j(p_i - u) > 0, & \text{when } j \leftarrow\leftarrow i, \\ 0, & \text{when } j \leftarrow i, \\ w_i u > 0, & \text{when } j \leftarrow i. \end{cases} \quad (3)$$

By using (2) and (3), it is easy to prove the following lemma which gives an expression for the increment from one solution to another.

Lemma 1. For two solutions (x^1, s^1) and (x^2, s^2) with the sets of crashed jobs $L(x^1, s^1)$ and $L(x^2, s^2)$, respectively, assume that $L(x^1, s^1) \subset L(x^2, s^2)$ and let $L(x^2, s^2) \setminus L(x^1, s^1) = \{k_1, k_2, \dots, k_q\}$. If $j \notin L(x^2, s^2)$ (and hence $j \notin L(x^1, s^1)$), then the increment due to $j \downarrow$ is

$$E_j(x^2, s^2) = E_j(x^1, s^1) + \sum_{i=1}^q \alpha_{k_i, j} \geq E_j(x^1, s^1).$$

Similarly, if $i \in L(x^1, s^1)$ (and hence $i \in L(x^2, s^2)$), then the increment due to $i \uparrow$ is

$$E_i(x^2, s^2) = E_i(x^1, s^1) - \sum_{i=1}^q \alpha_{k_i, j} \leq E_i(x^1, s^1).$$

For the special case where $q=1$, we have

- for solution (x, s) with $j \notin L(x, s)$, let (x', s') be the new solution due to $j \downarrow$, then

$$E_k(x', s') = \begin{cases} -E_j(x, s), & \text{if } k = j, \\ E_k(x, s) + \alpha_{jk}, & \text{if } k \neq j, \text{ and } k \notin L(x, s), \\ E_k(x, s) - \alpha_{jk}, & \text{if } k \in L(x, s); \end{cases} \quad (4)$$

- for solution (x, s) with $i \in L(x, s)$, let (x', s') be the new solution due to $i \uparrow$, then

$$E_k(x', s') = \begin{cases} -E_i(x, s), & \text{if } k = i, \\ E_k(x, s) - \alpha_{ik}, & \text{if } k \notin L(x, s), \\ E_k(x, s) + \alpha_{ik}, & \text{if } k \neq i, \text{ and } k \in L(x, s); \end{cases} \quad (5)$$

- for solution (x, s) with $j \notin L(x, s)$ and $i \in L(x, s)$, let (x', s') be the new solution due to $i \uparrow$ and $j \downarrow$, then

$$E_k(x', s') = \begin{cases} -E_j(x, s) + \alpha_{ij}, & \text{if } k = j, \\ -E_i(x, s) + \alpha_{ij}, & \text{if } k = i, \\ E_k(x, s) + \alpha_{jk} - \alpha_{ik}, & \text{if } k \neq j \text{ and } k \notin L(x, s), \\ E_k(x, s) - \alpha_{jk} + \alpha_{ik}, & \text{if } k \neq i \text{ and } k \in L(x, s). \end{cases} \quad (6)$$

And also, by (4) and (5), we have

$$F(x', s') - F(x, s) = E_i(x, s) + E_j(x, s) - \alpha_{ij}. \quad (7)$$

§ 3 The Main Theorem

In this section, we provide a necessary and sufficient condition for an optimal solution. Before we present the theorem, we give two lemmas, Lemma 2 and Lemma 4.

The following lemma and its corollary state that in the case of $i \leftarrow j$ or $i \leftarrow \leftarrow j$, we prefer crash i and resume j if i is not crashed and j is crashed.

- Lemma 2.** (a) If $i, j \notin L(x, s)$, and $i \leftarrow j$ or $i \leftarrow \leftarrow j$, then

$$E_i(x, s) \leq E_j(x, s) + \alpha_{ij} - w_i u \leq E_j(x, s).$$
 (b) If $i, j \in L(x, s)$, and $i \leftarrow j$ or $i \leftarrow \leftarrow j$, then

$$E_i(x, s) \geq E_j(x, s) - \alpha_{ij} + w_i u \geq E_j(x, s).$$

From Lemma 2 we can obtain the next corollary.

Corollary 3. For solution (x, s) with $i \notin L(x, s)$ and $j \in L(x, s)$, if $i \leftarrow j$ or $i \leftarrow \leftarrow j$, then the new solution (x', s') from (x, s) due to $i \downarrow$ and $j \uparrow$ is no worse than (x, s) , i. e. $F(x', s') \leq F(x, s)$.

Now for any solution (x, s) , by applying the above corollary for finite times we can obtain a new solution (\bar{x}, \bar{s}) which satisfies:

- (a) $F(\bar{x}, \bar{s}) \leq F(x, s)$, and
 (b) for any $i \notin L(\bar{x}, \bar{s})$ and $j \in L(\bar{x}, \bar{s})$, we have neither $i \leftarrow j$ nor $i \leftarrow \leftarrow j$.

Let D be the set of all such solutions (\bar{x}, \bar{s}) that satisfy (a) and (b). By the definition of dominance set given by Baker^[10], the set D is a dominance set. To solve our problem, we only need to find the best solution in the dominance set D . We call the solutions in D *dominance solutions*, and the optimal solution in D the *optimal dominance solution* which is the optimal solution to our problem.

For any pair i and j in a dominance solution (x, s) with $i \notin L(x, s)$ and $j \in L(x, s)$, only four possible cases should be considered: $j \leftarrow i, j \leftarrow \leftarrow i, j \leftarrow i$, and $i \leftarrow j$. Also by the definition of the notations " \leftarrow ", " $\leftarrow \leftarrow$ " and " $\leftarrow \leftarrow \leftarrow$ ", we can see that all crashed jobs are before all uncrashed jobs in a dominance solution, which approved the result we mentioned in Section 2. That is, all crashed jobs should be scheduled before any uncrashed job.

Lemma 4. Suppose that jobs $i_l \in L(x, s), l=1, 2, \dots, t$ and jobs $j_m \notin L(x, s), m=1, 2, \dots, h$, and

$$\frac{p_{i_1}}{w_{i_1}} \leq \frac{p_{i_2}}{w_{i_2}} \leq \dots \leq \frac{p_{i_t}}{w_{i_t}} \text{ and } \frac{p_{j_1} - u}{w_{j_1}} \leq \frac{p_{j_2} - u}{w_{j_2}} \leq \dots \leq \frac{p_{j_h} - u}{w_{j_h}}.$$

Let (x^1, s^1) be the solution obtained from (x, s) due to $j_1 \downarrow, j_2 \downarrow, \dots, j_{h-1} \downarrow, i_2 \uparrow, i_3 \uparrow, \dots, i_t \uparrow$, and (x^2, s^2) be the solution obtained from (x^1, s^1) due to $j_h \downarrow, i_1 \uparrow$. If for all $m=1, 2, \dots, h$ and $l=1, 2, \dots, t$, we have

$$j_m \leftarrow i_l \text{ or } i_l \leftarrow j_m, \text{ and } F(x^2, s^2) < F(x^1, s^1),$$

then there exist $j_m \in \{j_1, j_2, \dots, j_h\}, i_l \in \{i_1, i_2, \dots, i_t\}$, and $\{\bar{x}, \bar{s}\}$ obtained from (x, s) due to $j_m \downarrow$ and $i_l \uparrow$, such that $F(\bar{x}, \bar{s}) < F(x, s)$.

Now we give the main theorem of this paper.

Theorem 5. Let j_1, j_2, \dots, j_h and i_1, i_2, \dots, i_t be defined as in Lemma 4, and (\bar{x}, \bar{s}) be the solution obtained from (x, s) due to $j_1 \downarrow, j_2 \downarrow, \dots, j_h \downarrow$ and $i_1 \uparrow, i_2 \uparrow, \dots, i_t \uparrow$. If $F(\bar{x}, \bar{s}) < F(x, s)$, then at least one of the following situations is true:

- (a) there is $j_m \in \{j_1, j_2, \dots, j_h\}$ such that (x, s) can be strictly improved due to crashing j_m ;
- (b) there is $i_l \in \{i_1, i_2, \dots, i_t\}$ such that (x, s) can be strictly improved due to resuming i_l ;
- (c) there are $j_m \in \{j_1, j_2, \dots, j_h\}$ and $i_l \in \{i_1, i_2, \dots, i_t\}$ such that (x, s) can be strictly improved due to crashing j_m and resuming i_l .

Proof. If the set $\{i_1, i_2, \dots, i_t\}$ is empty, we prove by contradiction that the situation (a) in the theorem is true. Let (x^m, s^m) = the solution from (x, s) by $j_1 \downarrow, j_2 \downarrow, \dots, j_m \downarrow, m = 1, 2, \dots, h$. Hence $(x^h, s^h) = (\bar{x}, \bar{s})$, and $L(x, s) \subset L(x^1, s^1) \subset L(x^2, s^2) \subset \dots \subset L(x^h, s^h)$. If (a) not true, we have $F(x^1, s^1) \geq F(x, s)$. Also for any $m \geq 2, (x, s)$ can not be improved by crashing j_m . From Lemma 1, (x^{m-1}, s^{m-1}) can not be improved by crashing j_m either, i. e.

$$F(x^m, s^m) \geq F(x^{m-1}, s^{m-1}).$$

Therefore $F(\bar{x}, \bar{s}) = F(x^h, s^h) \geq F(x^{h-1}, s^{h-1}) \geq \dots \geq F(x^1, s^1) \geq F(x, s)$, which contradicts the assumption $F(\bar{x}, \bar{s}) < F(x, s)$ in the theorem.

If the set $\{j_1, j_2, \dots, j_h\}$ is empty, the situation (b) in the theorem can be similarly proved by contradiction.

If none of $\{i_1, i_2, \dots, i_t\}$ and $\{j_1, j_2, \dots, j_h\}$ is empty, assume that neither of the situations (a) or (b) is true. We show that (c) must be true by induction. When $t = h = 1, (c)$ is obviously true. We assume that (c) is true for all t and h with $t + h \leq N - 1$, and show that (c) is also true for t and h with $t + h = N$. Let (x^0, s^0) be the solution obtained from (x, s) by $j_1 \downarrow, j_2 \downarrow, \dots, j_{h-1} \downarrow$ and $i_2 \uparrow, i_3 \uparrow, \dots, i_t \uparrow$.

If $F(x^0, s^0) \geq F(x, s)$, solution (\bar{x}, \bar{s}) can be obtained from (x^0, s^0) by $j_h \downarrow$ and $i_1 \uparrow$, and (c) is then true by Lemma 4 and the assumption $F(\bar{x}, \bar{s}) < F(x, s) \leq F(x^0, s^0)$ in the theorem.

If $F(x^0, s^0) < F(x, s)$, which implies that $h \geq 2$ and $t \geq 2$, then the number of changed status of jobs in $j_1 \downarrow, j_2 \downarrow, \dots, j_{h-1} \downarrow, i_2 \uparrow, i_3 \uparrow, \dots, i_t \uparrow$ is $(h-1) + (t-1) = N - 2 < N - 1$, then (c) must be true by induction.

Corollary 6. Let (x, s) be a dominance solution. If (x, s) can not be improved by $j \downarrow, i \uparrow$, or $j \downarrow$ and $i \uparrow$ for any $i \in L(x, s)$ and $j \notin L(x, s)$, then (x, s) is optimal to the problem.

Proof. If not, let (x^0, s^0) be an optimal dominance solution to the problem, then $F(x^0, s^0) < F(x, s)$. Let $L(x^0, s^0) \setminus L(x, s) = \{j_1, j_2, \dots, j_h\}, L(x, s) \setminus L(x^0, s^0) = \{i_1, i_2, \dots, i_t\}$. By assumption, both (x^0, s^0) and (x, s) are dominance solutions. For any $m = 1, 2, \dots, h$ and $l = 1, 2, \dots, t$, we have either $j_m \leftarrow i_l$ or $i_l \leftarrow j_m$. Therefore (x^0, s^0) can be obtained from $(x,$

s) by $j_1 \downarrow, j_2 \downarrow, \dots, j_h \downarrow, i_1 \uparrow, i_2 \uparrow, \dots, i_t \uparrow$. By Theorem 5, at least one of the three situations is true, which contradicts the assumption in this corollary.

In Corollary 6 we give a necessary and sufficient condition for optimal dominance solutions, which theoretically supports the algorithm we will propose in the next section.

§ 4 Algorithm

In this section, we introduce an algorithm that is based on the theoretical results obtained in Section 3.

Before developing the algorithm, we introduce a definition of k -feasible solutions. We say a dominance solution is k -feasible if jobs $k+1, k+2, \dots, n$ have not been considered on crashing. Since we know that these $(n-k)$ uncrashed jobs are at the end of the sequence, a k -feasible solution is also a $(k+1)$ -feasible solution. We also call an optimal solution among all those k -feasible solutions a k -optimal solution. Then an n -optimal solution is an optimal solution to our problem. All lemmas, the theorem, and their corollaries in Section 3 still remain valid for k -feasible and k -optimal solutions.

The algorithm is the procedure to construct a $(k+1)$ -optimal solution based on the current k -optimal solution. We first give a theorem.

Theorem 7. Let $(x^*(k), s^*)$ be a k -optimal solution, $(x^0(k+1), s^0)$ be from $(x^*(k), s^*)$ due to $(k+1) \downarrow$, and $(x^l(k+1), s^l)$ due to $(k+1) \downarrow$ and $l \uparrow$, for $l \in L(x^*(k), s^*)$. Let

$$r = \min\{F(x^l(k+1), s^l) - F(x^*(k), s^*) \mid l \in \{0\} \cup L(x^*(k), s^*)\}. \tag{8}$$

If $r \geq 0$, we set $(x^*(k+1), s^*) = (x^*(k), s^*)$; If $r < 0$, we let

$$l = \max\{\bar{l} \mid r = F(x^{\bar{l}}(k+1), s^{\bar{l}}) - F(x^*(k), s^*), \bar{l} \in \{0\} \cup L(x^*(k), s^*)\}, \tag{9}$$

and set $(x^*(k+1), s^*) = (x^l(k+1), s^l)$.

Then the solution $(x^*(k+1), s^*)$ constructed in this way is $(k+1)$ -optimal.

When $l \in \{0\} \cup L(x^*(k), s^*)$, by (4) we have

$$F(x^l(k+1), s^l) - F(x^*(k), s^*) = E_{k+1}(x^*(k), s^*) + E_l(x^*(k), s^*) - \alpha_{l, k+1},$$

where we agree that $E_0(x, s) = 0, \alpha_{0j} = \alpha_{j0} = 0$. In short we also write $E_j^k = E_j(x^*(k), s^*), F^k = F(x^*(k), s^*)$, and $L^k = L(x^*(k), s^*)$, then (8) becomes

$$r = \min\{E_{k+1}^k + E_l^k - \alpha_{l, k+1} \mid l \in \{0\} \cup L^k\}, \tag{10}$$

and (9) becomes

$$l = \max\{\bar{l} \mid r = E_{k+1}^k + E_{\bar{l}}^k - \alpha_{\bar{l}, k+1}, \bar{l} \in \{0\} \cup L^k\}. \tag{11}$$

The following algorithm is based on Theorem 7.

Algorithm

Step 1. Originally, jobs are ordered such that $\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \dots \leq \frac{p_n}{w_n}$, and $p_j \leq p_{j+1}$ when

$\frac{p_j}{w_j} = \frac{p_{j+1}}{w_{j+1}}$, i. e. $1 \leftarrow 2 \leftarrow \dots \leftarrow n$. Calculate $F^0 = F(0, s^0)$, where $0 = (0, 0, \dots, 0)$ and $s^0 =$

$(1, 2, \dots, n)$. Compute $E_j^0 = E_j(0, s^0)$ by using (2), α_{ij} by using (3), for $i, j = 1, 2, \dots, n$. Also we agree that $E_0^0 = 0, E_0^j = 0, \alpha_{0j} = \alpha_{j0} = 0$, for $j = 1, 2, \dots, n, L^0 = \emptyset$. Let $k = 0$.

Step 2. If $k = n$, the algorithm stops. F^n is the optimal cost. Crash all jobs in L^n to obtain x , as well as the corresponding WSPT sequence s . (x, s) is the optimal solution. If $k < n$, compute $r = \min\{E_{k+1}^k + E_l^k - \alpha_{l, k+1} \mid l \in \{0\} \cup L^k\}$.

Step 3. If $r \geq 0$, then set $E_j^{k+1} = E_j^k, j = 1, 2, \dots, n, L^{k+1} = L^k$, and $F^{k+1} = F^k$. Let $k = k + 1$. Go to Step 2.

If $r < 0$, for $l = \max\{\bar{l} \mid r = E_{k+1}^k + E_{\bar{l}}^k - \alpha_{l, k+1}, \bar{l} \in \{0\} \cup L^k\}$, we set $F^{k+1} = F^k + r$, and if $l = 0$, then by (4) we set

$$E_m^{k+1} = \begin{cases} -E_{k+1}^k, & \text{if } m = k + 1. \\ E_m^k + \alpha_{m, k+1}, & \text{if } m \neq k + 1, m \notin L^k, \\ E_m^k - \alpha_{m, k+1}, & \text{if } m \in L^k. \end{cases}$$

Let $L^{k+1} = L^k \cup \{k+1\}$, and $k = k + 1$. Go to Step 2. if $l > 0$, then by (6) we set

$$E_m^{k+1} = \begin{cases} -E_{k+1}^k - \alpha_{l, k+1}, & \text{if } m = k + 1, \\ -E_l^k + \alpha_{l, k+1}, & \text{if } m = l, \\ E_m^k + \alpha_{m, k+1} - \alpha_{lm}, & \text{if } m \neq k + 1, m \notin L^k, \\ E_m^k - \alpha_{m, k+1} + \alpha_{lm}, & \text{if } m \neq l, m \in L^k. \end{cases}$$

Let $L^{k+1} = L^k \cup \{k+1\} \setminus \{l\}$, and $k = k + 1$. Go to Step 2.

The complexity of the our algorithm is $O(n^2)$.

§ 5 A Numerical Example

In order to illustrate the algorithm proposed in Section 4, we consider an example with the following data.

Table $u = 1, c = 8$

Job j	1	2	3	4
p_j	6	7.5	7.5	1.9
w_j	5	4	4	1

Initially, we have $x^0 = (0, 0, 0, 0), s^0 = (1, 2, 3, 4), L^0 = \emptyset$ and $F^0 = F(x^0, s^0) = 190.9$.

$k = 0; r = -6 < 0, l = 0, F^1 = F^0 + r = 184.9$ and upgrade E_j^1 's. Crash job 1 and obtain $L^1 = \{1\}$.

$k = 1; r = -1 < 0, l = 0, F^2 = F^1 + r = 183.9$ and upgrade E_j^2 's. Crash job 2 and obtain $L^2 = L^1 \cup \{2\} = \{1, 2\}$.

$k = 2; \text{Since } r = 0, F^3 = F^2 = 183.9$ and $L^3 = L^2 = \{1, 2\}$.

$k = 3; r = -0.3 < 0, l = 2, F^4 = F^3 + r = 183.6$ and upgrade E_j^4 's. Crash job 4, resume job 2 and $L^4 = L^3 \cup \{4\} \setminus \{2\} = \{1, 4\}$.

The optimal crash decision is $x^* = (1, 0, 0, 1)$, and the optimal sequence is the corre-

sponding WSPT sequence which is $s^* = (4, 1, 2, 3)$. The optimal cost is 183.6.

§ 6 Conclusion

In this paper, we have considered the scheduling problem with controllable processing times and compression costs. A special case of equal controllable processing times and equal compression costs is discussed. We provide analytical results and developed an algorithm to solve the special case optimally. There are several future research directions. First, nonlinear compression cost may be considered as an extension. In fact, it is very possible that the compression cost is a lump sum (the special case discussed in this paper) or a convex / concave function of the time crashed depending on how the resources are used. Second, incorporating multiple compression costs might be an interesting extension since compressing a job usually requires multiple resources: staff, facility, and materials. To aggregate costs of several resources in crashing is one way. To separate them to reflect their different requirements in compression is another way. Finally, a more challenging extension might be to consider multiple identical machines. This extension, in fact, is more realistic and more applicable in the real sphere.

References

- [1] Lawler, E. L. and Morre, J. M. , A functional equation and its application to resource allocation and sequencing problems, *Management Science*, **16**(1969), 77-84.
- [2] Vickson, R. G. , Choosing the job sequence and processing times to minimize total processing plus flow cost on single machine, *Oper. Res.* , **28**(1980), 1155-1167.
- [3] Vickson, R. G. , Two single-machines sequencing problems involving controllable job processing times, *IIE Trans.* , **12**(1980), 258-262.
- [4] Van Wassenhove, L. N. and Baker, K. R. , A bicriterion approach to time/cost tradeoffs in sequencing, *European Journal of Operational Research*, **11**(1982), 48-54.
- [5] Tuzikov, A. V. , A two-criterial scheduling problem allowing for variation in job execution, *Zh. Vychisl. Mat. i Mat. Fiz.* , **24**(1984), 1585-1590.
- [6] Ruiz Diaz, F. M. and French, S. , A survey of multi-objective combinatorial scheduling, In: French, S. , Hartley, R. , Thomas, L. C. , and White, D. J. , eds. , *Multi-Objective Decision Making*, Academic Press, New York, 1983.
- [7] Nowicki, E. and Zdrzalka, S. , A survey of results for sequencing problems with controllable processing times, *Discrete Applied Mathematics*, **26**(1990), 271-287.
- [8] Tang Guochun, Foulds, L. R. , Single machine scheduling with controllable processing times and compression costs: proof of theorems, *Appl. Math. - JCU*, **13B**(1998), 427-436.
- [9] Smith, W. E. , Various optimizers for single-stage production, *Naval Res. Logist.* , **3**(1956), 59-66.
- [10] Baker, K. R. , *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.

Department of Management, Shanghai Second Polytechnic University, Shanghai 200002.

Department of Management Systems, University of Waikato, Hamilton, New Zealand.