Single Machine Scheduling with Resource Dependent Release Times and Processing Times

Xiuli Wang[a] T. C. E. Cheng[b#]

[a]Department of Automation
Shanghai Jiaotong University
Shanghai, P. R. China

[b]Department of Logistics
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
E-mail: LGTcheng@polyu.edu.hk

Abstract

We consider the single machine scheduling problem with resource dependent release times and processing times, in which both the release times and processing times are strictly linear decreasing functions of the amount of resources consumed. The objective is to minimize the makespan plus the total resource consumption costs. We propose a heuristic algorithm for the general problem by utilizing some derived optimal properties and analyze its performance bound. For some special cases, we propose another heuristic algorithm that achieves a tighter performance bound.

Keywords: Scheduling, resource dependent release times, resource dependent processing times, heuristics, performance bound.

1. Introduction

The scheduling problem with resource dependent processing times has received much research attention in recent years. Studies in this area were initiated by Vickson

Corresponding author.

[15, 16] and Van Wassenhove and Baker [14]. A survey of this topic up to 1990 was given by Nowicki and Zdrzalka [11]. During the last decade, some new results on these problems have appeared in the literature. They can be found in Zdrzalka [17], Panwalkar and Rajagopalan [13], Alidaee and Ahmadian [1], Nowicki and Zdrzalka [12], Cheng et al. [3], Janiak and Kovalyov [9], Chen et al. [2], Cheng et al. [5] and Zhang et al. [18], among others. The scheduling models cited above all assume that each job is available at the beginning or its release time is constant.

The scheduling problem with resource dependent release times has also received considerable attention of the scheduling research community in recent years. Some research results can be found in the following papers: Janiak [7, 8], Cheng and Janiak [4], Cheng and Kovalyov [6] and Li et al. [10], among others. In these scheduling models, the jobs are each assumed to have a fixed processing time.

However, to the best of our knowledge, there seem to exist no papers studying the scheduling problem in which both release times and processing times are resource dependent. Such a scheduling problem commonly arises in the chemical processing industry. Before chemical compounds (jobs) are ready for processing, they have to be preheated to reach a temperature threshold below which chemical reactions will not take place. This preheating process consumes resources such as fuel and so a chemical compound is ready earlier for processing if more fuel is consumed to preheat it. On the other hand, the processing time of a chemical compound varies according to the speed of its chemical reaction, which is directly related to the amount of catalysts consumed. Hence, both the job release times and processing times are variable and depend on the amount of resources consumed. The objective of the scheduling problem is to minimize the sum of resource consumption and the makespan, i.e., the total elapsed time to complete all jobs. Such a situation can be modeled as our scheduling problem with resource dependent release times and processing times on a single machine.

This paper is organized as follows. In Section 2, we formulate the problem under study. In Section 3, we derive some properties of an optimal solution. In Section 4, we present a heuristic algorithm for the general problem and analyze its performance

bound. In Section 5, we present a heuristic algorithm for some special cases that yields a tighter performance bound. Section 6 concludes with a summary and suggestions for further research.

2. Problem Formulation

In this section the single machine scheduling problem is considered under the assumption that both release times and processing times are strictly linear decreasing functions of the amount of resources consumed. Formally, the problem can be formulated as follows.

We consider the problem of scheduling a set $J = \{J_1, \ldots, J_n\}$ of n jobs on a single machine. Let π denote a permutation of the jobs in set J and Π the set of all such permutations. All jobs are initially available at time v, but each job may be made available at an earlier time point by consuming extra resources (e.g. fuel) that will incur additional costs. Associated with each job J_i is a processing time p_i and a release time r_i, $i = 1, 2, \ldots, n$, where both p_i and r_i depend on the amount of resources consumed. Specifically, $p_i = a_i - x_i$, where a_i is the normal processing time and x_i the amount of processing time compression, $0 \le x_i \le a_i$; $r_i = v - u_i / w$, where u_i is the amount of resource consumed to advance the availability of J_i to r_i and w the cost per unit reduction of release time, $0 \le u_i \le wv$. For the convenience of theoretically analyzing the studied problem, it should be noted that the case $p_i = 0$ means that the actual processing time of job J_i is so small that it can be neglected in the objective function. Let $x = (x_1, x_2, \cdots, x_n)$ denote a processing time compression vector and $r = (r_1, r_2, \cdots, r_n)$ a release time vector. Also, let X denote the set of all feasible x and R the set of all feasible r.

For any job J_i, since the amount of resources consumed for its release time reduction u_i, $u_i = w(v - r_i)$, is a decreasing function of the release time r_i, $i = 1, 2, \ldots, n$, we assume that all jobs start as early as possible after they are released. For given π,

x and r, assuming the job permutation $\pi = (J_1, \ldots, J_n)$, the objective function, or total cost, $K(x, r, \pi)$ is defined as

$$K(x,r,\pi) = \max_{1\le j\le n}\{r_j + \sum_{i=j}^{n}(a_i - x_i)\} + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} w(v - r_i), \quad (1)$$

where c_i, $i = 1, 2, \ldots, n$, is the cost per unit processing time reduction. To simplify notation, we assume that both c_i , $0 < c_i < 1$, and w, $0 < w < 1$, are appropriately scaled so that their units are compatible with that of the makespan. It is clear that $\max_{1\le j\le n}\{r_j + \sum_{i=j}^{n}(a_i - x_i)\}$ is the makespan, $\sum_{i=1}^{n} c_i x_i$ is the total processing time compressing cost, and $\sum_{i=1}^{n} w(v - r_i)$ is the total release time compressing cost. Thus, the optimal objective $K^*(x, r, \pi)$ is

$$K^*(x,r,\pi) = \min_{x\in X,\, r\in R,\, \pi\in\Pi} K(x,r,\pi). \quad (2)$$

Under the constraint of a common deadline, the single machine scheduling problem to minimize resource consumption, in which the job release times follow a linear model, $r_i = v - u_i / w,\ i = 1, 2, \cdots, n$, while the processing times are constant, is NP-hard in the ordinary sense (Janiak [8]). In our problem the release times follow the same linear model as that in Janiak [8], but the processing times are also linearly dependent on the amount of resources the jobs have consumed. So the problem studied here is more difficult to deal with, and is evidently NP-hard. Thus, we will focus on developing heuristic algorithms for the problem under study.

3. Problem Analysis

In this section we establish some properties of an optimal solution to the scheduling problem under consideration.

We first note that if there exists idle time between the first job and the last job in a permutation π, the total cost can be reduced by eliminating the idle time through changing the actual processing times and release times of some jobs. So we have the following lemma.

Lemma 1 In an optimal schedule, there exists no idle time between the first and the last processed jobs.

Proof In a given solution (x, r, π) with an objective function $K(x, r, \pi)$, if there exists an integer m ($0 < m < n$) such that the idle time between the mth job and (m+1)st job is Δt ($\Delta t > 0$), we can construct a new solution (x, r', π) with an objective function $K(x, r', \pi)$ such that $K(x, r', \pi) < K(x, r, \pi)$.

If $v \geq r_m + (a_m - x_m) + \Delta t$, we set $r' = (r_1', r_2', \cdots, r_n') = (r_1 + \Delta t, \cdots, r_m + \Delta t, r_{m+1}, \cdots, r_n)$. Then $K(x, r', \pi)$ is

$$\begin{aligned} K(x, r', \pi) &= \max_{1 \leq j \leq n}\{r_j' + \sum_{i=j}^{n}(a_i - x_i)\} + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} w(v - r_i') \\ &= \max_{1 \leq j \leq n}\{r_j + \sum_{i=j}^{n}(a_i - x_i)\} + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} w(v - r_i) - m\Delta t \\ &= K(x, r, \pi) - m\Delta t < K(x, r, \pi). \end{aligned}$$

If $r_m + (a_m - x_m) < v < r_m + (a_m - x_m) + \Delta t$, we set $\Delta t = \Delta t_1 + \Delta t_2$, where $r_m + (a_m - x_m) + \Delta t_1 = v$, and $r' = (r_1', r_2', \cdots, r_n') = (r_1 + \Delta t_1, \cdots, r_m + \Delta t_1, r_{m+1} - \Delta t_2, \cdots, r_n - \Delta t_2)$. Then $K(x, r', \pi)$ is

$$\begin{aligned} K(x, r', \pi) &= \max_{1 \leq j \leq n}\{r_j' + \sum_{i=j}^{n}(a_i - x_i)\} + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} w(v - r_i') \\ &= \max_{1 \leq j \leq n}\{r_j + \sum_{i=j}^{n}(a_i - x_i)\} - \Delta t_2 + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} w(v - r_i) - m\Delta t_1 \\ &= K(x, r, \pi) - (m\Delta t_1 + \Delta t_2) < K(x, r, \pi). \end{aligned}$$

If $v < r_m + (a_m - x_m)$, we set $r' = (r_1', r_2', \cdots, r_n') = (r_1, \cdots, r_m, r_{m+1} - \Delta t, \cdots, r_n - \Delta t)$. Then $K(x, r', \pi)$ is

$$\begin{aligned} K(x, r', \pi) &= \max_{1 \leq j \leq n}\{r_j' + \sum_{i=j}^{n}(a_i - x_i)\} + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} w(v - r_i') \\ &= \max_{1 \leq j \leq n}\{r_j + \sum_{i=j}^{n}(a_i - x_i)\} - \Delta t + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} w(v - r_i) \\ &= K(x, r, \pi) - \Delta t < K(x, r, \pi). \end{aligned}$$

Summarizing the above discussion, the conclusion holds. □

According to Lemma 1, the objective function $K(x, r, \pi)$ can be re-written as

$$K(x,r,\pi) = r_1 + \sum_{i=1}^{n}(a_i - x_i) + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} w(v - r_i). \qquad (3)$$

Further, although an optimal solution is determined by the factors π, x and r, these factors are interrelated in the process of searching for an optimal solution. So it is necessary to analyze the relations among these factors in an optimal solution. In the following, we derive some optimal properties to determine the release times for given π and x. We also develop some optimal properties to sequence jobs and determine the processing time compressions simultaneously.

In an optimal solution, let π^* denote its permutation and x^* its processing time compression vector. We have

Lemma 2 For π^* and x^* in an optimal solution, r_i, i= 2, …, n, should satisfy the following conditions:

i) If $r_1 + \sum_{j=1}^{i-1}(a_j - x^*_j) \le v$, then $r_i = r_1 + \sum_{j=1}^{i-1}(a_j - x^*_j)$;

ii) If $r_1 + \sum_{j=1}^{i-1}(a_j - x^*_j) > v$, then $r_i = v$.

Proof i) If $r_i < r_1 + \sum_{j=1}^{i-1}(a_j - x^*_j)$, we set $\Delta r_i = r_1 + \sum_{j=1}^{i-1}(a_j - x^*_j) - r_i$, and $\bar{r} = (r_1, \cdots, r_{i-1}, r_i + \Delta r_i, r_{i+1}, \cdots, r_n)$. From (3), we have

$$K(x^*, \bar{r}, \pi^*) - K(x^*, r, \pi^*) = w[v - (r_i + \Delta r_i)] - w(v - r_i) = -w\Delta r_i < 0.$$

If $r_i > r_1 + \sum_{j=1}^{i-1}(a_j - x^*_j)$, it contradicts Lemma 1.

ii) The conclusion holds trivially. □

We assume that the constant w is such that $\frac{1}{k} < w \le \frac{1}{k-1}$, where k is an integer and $k \ge 2$. We will notice in the sequel that, to minimize the objective function, the

optimal determination of the release time vector is directly affected by the integer *k*.

For given permutation π and $x = (x_1, x_2, \cdots, x_n)$, in view of the optimal properties of Lemmas 1 and 2, we can determine $r = (r_1, r_2, \cdots, r_n)$ using the following algorithm.

Algorithm A1

Step 1. If $p_1 + \cdots + p_{k-1} > v$ holds, then set $r_1 = 0$; otherwise, set

$$r_1 = v - (p_1 + \cdots + p_{k-1}).$$

Step 2. If $r_1 = 0$, then determine the integer m, $m < k$, such that $p_1 + \cdots + p_{m-1} \le v < p_1 + \cdots + p_m$ holds. Set $r_2 = p_1$, $r_3 = p_1 + p_2, \cdots, r_m = p_1 + \cdots + p_{m-1}$, $r_{m+1} = v, \cdots, r_n = v$. Otherwise, set $r_2 = r_1 + p_1$, $r_3 = r_1 + p_1 + p_2, \cdots, r_{k-1} = r_1 + p_1 + \cdots + p_{k-2}$, $r_k = r_1 + p_1 + \cdots + p_{k-1} = v, r_{k+1} = v, \cdots, r_n = v$.

According to Algorithm A1, for the case where $p_1 + \cdots + p_{k-1} > v$, the objective function $\overline{K}_1(x, r, \pi)$ is

$$\begin{aligned}
\overline{K}_1(x, r, \pi) &= \sum_{i=1}^{n}(a_i - x_i) + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} w(v - r_i) \\
&= \sum_{i=1}^{n}(a_i - x_i) + \sum_{i=1}^{n} c_i x_i + wv + w[v - (a_1 - x_1)] + \cdots + w[v - \sum_{i=1}^{m-1}(a_i - x_i)] \\
&= \sum_{i=1}^{n}(a_i - x_i) + \sum_{i=1}^{n} c_i x_i + w[\sum_{i=1}^{m-1}(a_i - x_i) + d_m] + w[\sum_{i=2}^{m-1}(a_i - x_i) + d_m] \\
&\quad + \cdots + w[(a_{m-1} - x_{m-1}) + d_m] + wd_m \\
&= \sum_{i=1}^{n}(a_i - x_i) + \sum_{i=1}^{n} c_i x_i + mwd_m + w[(a_1 - x_1) + 2(a_2 - x_2) \\
&\quad + \cdots + (m-1)(a_{m-1} - x_{m-1})], \qquad (4)
\end{aligned}$$

where $d_m = v - \sum_{i=1}^{m-1}(a_i - x_i)$ and $0 \le d_m < a_m - x_m$.

For the case where $p_1 + \cdots + p_{k-1} \le v$, the objective function $\overline{K}_2(x, r, \pi)$ is

$$\begin{aligned}\overline{K}_2(x,r,\pi) &= r_1 + \sum_{i=1}^{n}(a_i - x_i) + \sum_{i=1}^{n} c_i x_i + w[(a_1 - x_1) + 2(a_2 - x_2) \\ &\quad + \cdots + (k-1)(a_{k-1} - x_{k-1})] \\ &= v + \sum_{i=k}^{n}(a_i - x_i) + \sum_{i=1}^{n} c_i x_i + w[(a_1 - x_1) + 2(a_2 - x_2) \\ &\quad + \cdots + (k-1)(a_{k-1} - x_{k-1})]. \end{aligned} \tag{5}$$

In Step 1 of Algorithm A1, we select the value of r_1 based on the following consideration.

If $p_1 + \cdots + p_{k-1} > v$, and if we set $0 < r_1 \le v$, then r_1 may be denoted as $r_1 = \Delta t_1 + \cdots + \Delta t_{m-1} + d'_m$, where $0 \le \Delta t_i \le p_i$, $i = 1, 2, \cdots, m-1$, $0 < d'_m \le d_m$. And assume that if $d'_m < d_m$, then $\Delta t_{m-1} = 0$, and if $\Delta t_i < p_i$ $(i = 2, 3, \cdots, m-1)$, then $\Delta t_h = 0$ for $h = 1, 2, \cdots, i-1$. According to Lemmas 1 and 2, $r_2 = r_1 + p_1 - \Delta t_1$, $r_3 = r_1 + p_1 + p_2 - (\Delta t_1 + \Delta t_2), \cdots, r_m = r_1 + p_1 + \cdots + p_{m-1} - (\Delta t_1 + \cdots + \Delta t_{m-1}), r_{m+1} = v, \cdots, r_n = v$. Then the objective function $K'(x, r, \pi)$ is

$$\begin{aligned} K'(x,r,\pi) &= r_1 + \sum_{i=1}^{n}(a_i - x_i) + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} w(v - r_i) \\ &= \overline{K}_1(x,r,\pi) + (1-w)\Delta t_1 + (1-2w)\Delta t_2 + \cdots + (1-(m-1)w)\Delta t_{m-1} \\ &\quad + (1\text{-}mw)d'_m \ge \overline{K}_1(x,r,\pi). \end{aligned} \tag{6}$$

If $p_1 + \cdots + p_{k-1} \le v$, and if we set $0 < r_1 < v - (p_1 + \cdots + p_{k-1})$, then r_1 may be denoted as $r_1 = v - (p_1 + \cdots + p_{k-1}) - \Delta t$, where $0 < \Delta t < v - (p_1 + \cdots + p_{k-1})$. According to Lemmas 1 and 2, $r_2 = r_1 + p_1, \cdots, r_{k-1} = r_1 + (p_1 + \cdots + p_{k-2}), r_k = v - \Delta t, r_{k+1} = v, \cdots, r_n = v$. Then the objective function $K'(x, r, \pi)$ is

$$\begin{aligned} K'(x,r,\pi) &= r_1 + \sum_{i=1}^{n}(a_i - x_i) + \sum_{i=1}^{n} c_i x_i + w[(a_1 - x_1) + 2(a_2 - x_2) \\ &\quad + \cdots + (k-1)(a_{k-1} - x_{k-1}) + k\Delta t] \\ &= v + \sum_{i=k}^{n}(a_i - x_i) + \sum_{i=1}^{n} c_i x_i + w[(a_1 - x_1) + 2(a_2 - x_2) \\ &\quad + \cdots + (k-1)(a_{k-1} - x_{k-1})] + (kw-1)\Delta t \\ &= \overline{K}_2(x,r,\pi) + (kw-1)\Delta t > \overline{K}_2(x,r,\pi). \end{aligned} \tag{7}$$

If $p_1 + \cdots + p_{k-1} \le v$, and if we set $v - (p_1 + \cdots + p_{k-1}) < r_1 \le v$, then r_1 may be denoted as $r_1 = v - (p_1 + \cdots + p_{k-1}) + (\Delta t_1 + \cdots + \Delta t_{k-1})$, where $0 < \Delta t_{k-1} \le p_{k-1}$, $0 \le \Delta t_i \le p_i, i = 1, 2, \cdots, k-2$. And assume that if $\Delta t_i < p_i$ $(i = 2, 3, \cdots, k-1)$, then $\Delta t_h = 0$ for $h = 1, 2, \cdots, i-1$. According to Lemmas 1 and 2, $r_2 = r_1 + p_1 - \Delta t_1$, $r_3 = r_1 + p_1 + p_2 - (\Delta t_1 + \Delta t_2)$, $\cdots$, $r_{k-1} = r_1 + (p_1 + \cdots + p_{k-2}) - (\Delta t_1 + \cdots + \Delta t_{k-2})$, $r_k = v, \cdots, r_n = v$. Then the objective function $K'(x, r, \pi)$ is

$$
\begin{aligned}
K'(x,r,\pi) &= r_1 + \sum_{i=1}^{n}(a_i - x_i) + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} w(v - r_i) \\
&= v + \sum_{i=k}^{n}(a_i - x_i) + w[(a_1 - x_1) + 2(a_2 - x_2) + \cdots + (k-1)(a_{k-1} - x_{k-1})] \\
&\quad + (1-w)\Delta t_1 + (1-2w)\Delta t_2 + \cdots + (1-(k-1)w)\Delta t_{k-1} \\
&= \overline{K}_2(x,r,\pi) + (1-w)\Delta t_1 + (1-2w)\Delta t_2 + \cdots + (1-(k-1)w)\Delta t_{k-1} \\
&\ge \overline{K}_2(x,r,\pi). \qquad (8)
\end{aligned}
$$

From (6), (7) and (8), it is evident that the search for an optimal solution can be restricted to the space of the set (x, π) with r determined by Algorithm A1. According to Lemma 1, Lemma 2 and the above discussion, we have the following result.

Theorem 1 In an optimal solution, $r = (r_1, r_2, \cdots, r_n)$ should be determined by Algorithm A1.

In view of Theorem 1, we will denote $\overline{K}_1(x,r,\pi)$, $\overline{K}_2(x,r,\pi)$ and $K^*(x,r,\pi)$ by $\overline{K}_1(x,\pi)$, $\overline{K}_2(x,\pi)$ and $K^*(x,\pi)$, respectively. Thus, the optimal objective value $K^*(x,\pi)$ is given by

$$K^*(x,\pi) = \min_{x \in X, \pi \in \Pi}\{\overline{K}_1(x,\pi), \overline{K}_2(x,\pi)\}.$$

From the above discussion, we may focus our attention on analyzing the relation between a permutation π and a processing time compression vector x in order to search for an optimal schedule. In fact, the two factors π and x are interrelated in an

optimal solution. Next, we develop an optimal property to determine x under a given π.

For a given $\pi = (J_1, J_2, \ldots, J_n)$, say, to minimize the objective function $\overline{K}_1(x,\pi)$, we can treat the processing time compressions x_i, $i = 1, 2, \ldots, n$, as decision variables and formulate the problem as a linear programming problem P1:

(P1)

$$\begin{aligned}
&\min \overline{K}_1(x,\pi) \\
&\text{subject to} \\
&\quad x_i \le a_i,\ i = 1, 2, \cdots, n \\
&\quad x_1 + x_2 + \cdots + x_{m-1} = (a_1 + \cdots + a_{m-1}) - (v - d_m) \\
&\quad x_i \ge 0,\ i = 1, 2, \cdots, n.
\end{aligned}$$

Similarly, to minimize the objective function $\overline{K}_2(x,\pi)$, we can treat the processing time compressions x_i, $i = 1, 2, \ldots, n$, as decision variables and formulate the problem as a linear programming problem P2:

(P2)

$$\begin{aligned}
&\min \overline{K}_2(x,\pi) \\
&\text{subject to} \\
&\quad x_i \le a_i,\ i = 1, 2, \cdots, n \\
&\quad x_1 + x_2 + \cdots + x_{k-1} = (a_1 + \cdots + a_{k-1}) - (v - r_1) \\
&\quad x_i \ge 0, i = 1, 2, \cdots, n.
\end{aligned}$$

For problem P1, to minimize $\overline{K}_1(x,\pi)$, it is obvious from (4) that if $i \ge m$, J_i must be fully compressed. Moreover, since the coefficient of x_i , $i = 1, 2, \ldots, m$-1, in (4) is $-(1 - c_i + iw)$, and under the constraint that $x_1 + \cdots + x_n = (a_1 + \cdots + a_{m-1}) - (v - d_m)$, a job J_i whose processing time compression x_i has a larger negative coefficient must be fully compressed to minimize $\overline{K}_1(x,\pi)$. Specifically, the optimal solution is

$$x_i = \begin{cases} a_i & \text{if } J_i \in A_1 \text{ or } i \ge m \\ 0 & \text{if } J_i \in A_2 \\ \tilde{a}_s & \text{if } J_i \in A_3 \end{cases}, \qquad (9)$$

where $\tilde{a}_s \le a_s$, $1 \le s \le m-1$, $J_s \in A_3$, $\tilde{a}_s = (v - d_m) - \sum_{J_i \in A_2} a_i$, A_3 either consists of

only one job or is empty, $A_1 \bigcap A_2 = \Phi$ (i.e., the empty set), $A_1 \bigcup A_2 \bigcup A_3 = \{J_1, J_2, \cdots, J_{m-1}\}$, and for any $J_i \in A_2$ and $J_j \in A_1$, $1 - c_i + iw \le 1 - c_s + sw \le 1 - c_j + jw$ holds. Index s can be uniquely determined by sequencing the coefficients of $x_i (i = 1, 2, \cdots, m-1)$ in descending order and considering the constraints of $x_1 + \cdots + x_{m-1} = (a_1 + \cdots + a_{m-1}) - (v - d_m)$ and $0 \le x_i \le a_i$. The last condition helps to find the job J_s and it indicates that to minimize $\overline{K}_1(x, \pi)$, the jobs in A_1 are fully compressed while those in A_2 are not compressed. Given such choices of decision variables, the optimal objective value $\overline{K}_1 *(x, \pi)$ is

$$\overline{K}_1 *(x, \pi) = v + \sum_{J_i \in J \setminus (A_2 \cup A_3)} c_i a_i + (mw - 1) d_m + w \sum_{J_i \in A_2} i a_i + c_s \tilde{a}_s + sw(a_s - \tilde{a}_s). \quad (10)$$

For problem P2, by a similar argument, we see that the optimal solution is

$$x_i = \begin{cases} a_i & \text{if } J_i \in B_1 \text{ or } i \ge k \\ 0 & \text{if } J_i \in B_2 \\ \tilde{a}_s & \text{if } J_i \in B_3 \end{cases}, \quad (11)$$

where $\tilde{a}_s \le a_s, 1 \le s \le k-1, J_s \in B_3, \tilde{a}_s = v - \sum_{J_i \in B_2} a_i$, B_3 either consists of only one job or is empty, $B_1 \bigcap B_2 = \Phi$, $B_1 \bigcup B_2 \bigcup B_3 = \{J_1, J_2, \cdots, J_{k-1}\}$, and for any $J_i \in B_2$ and $J_j \in B_1$, $iw - c_i \le sw - c_s \le jw - c_j$ holds. Index s can be uniquely determined by sequencing the coefficients of $x_i (i = 1, 2, \cdots, k-1)$ in descending order and considering the constraints of $x_1 + \cdots + x_{k-1} = (a_1 + \cdots + a_{k-1}) - (v - r_1)$ and $0 \le x_i \le a_i$. Given such choices of decision variables, the optimal objective value $\overline{K}_2 *(x, \pi)$ is

$$\overline{K}_2 *(x, \pi) = v + \sum_{J_i \in J \setminus (B_2 \cup B_3)} c_i a_i + w \sum_{J_i \in B_2} i a_i + c_s \tilde{a}_s + sw(a_s - \tilde{a}_s). \quad (12)$$

Summarizing the above discussion, we have established the following theorem.

Theorem 2 In an optimal solution, the job processing time compression vector x

should be determined by (9) or (11).

Before sequencing jobs in set J, we can determine the compression of some jobs according to their unit compression cost c_i, i = 1, 2, …, n. In order to analyze such properties for a given problem, we divide set J into k subsets:

$$N_l = \{J_i \mid lw \le c_i < (l+1)w,\ J_i \in J\},$$

where l = 0, 1, …, k – 1. For a given problem, some of these subsets may be empty. From (4) and (5), we can easily obtain the following result.

Theorem 3 In an optimal solution, a job J_i belonging to N_0 should be fully compressed, i.e., if $J_i \in N_0$, then $x_i = a_i$.

Proof In a solution (x, r, π) with objective function (4) or (5), assume that $c_h < w$, i.e., $J_h \in N_0$. If we set $x_h = a_h$, for the case $p_1 + \cdots + p_{k-1} > v$, denote the objective function as $K_1'(x, r, \pi)$, and for the case $p_1 + \cdots + p_{k-1} \le v$, denote the objective function as $K_2'(x, r, \pi)$.

i) $p_1 + \cdots + p_{k-1} > v$.

1) If $h > m$, set $x_h = a_h$, and we have

$$K_1'(x, r, \pi) = \overline{K}_1(x, r, \pi) - (1 - c_h)(a_h - x_h) < \overline{K}_1(x, r, \pi) \quad .$$

2) If $h \le m$, set $x_h = a_h$, and for the two cases $\sum_{i=1}^{k} p_i - p_h > v$ and $\sum_{i=1}^{k} p_i - p_h \le v$, we can derive the following formulation from (4) and (5).

$$\begin{aligned} K_1'(x, r, \pi) &\le \overline{K}_1(x, r, \pi) - (hw - c_h)(a_h - x_h) - (1 - (k-1)w)(a_h - x_h) \\ &< \overline{K}_1(x, r, \pi). \end{aligned}$$

ii) $p_1 + \cdots + p_{k-1} \le v$.

In this case, similar to i), we can also obtain

$$K_2'(x, r, \pi) < \overline{K}_2(x, r, \pi).$$

Therefore, the conclusion holds. أ

4. A Heuristic Algorithm

Making use of the optimal properties derived in Section 3, we develop the following heuristic algorithm for the general problem.

Heuristic Algorithm HA1

Step 1. If $N_0 \neq \Phi$, for all jobs $J_i \in N_0$, set $x_i = a_i$; otherwise, go to Step 2.

Step 2. Let $T = \Phi$, $h = 0$, and s be the largest index such that $N_s \neq \Phi$ and $N_{s+1} = \Phi$.

For $g = 1$ to s, repeat:

If $N_g \bigcup \cdots \bigcup N_s \neq \Phi$, then let $a_g = \max\{a_i \mid J_i \in N_g \bigcup \cdots \bigcup N_s\}$.

If $\sum_{J_i \in T} a_i + a_g > v$, go to Step 3.

Otherwise, if $J_g \in N_m$, then let $N_m = N_m \setminus \{J_g\}$ and $T = T \bigcup \{J_g\}$, $h \leftarrow h+1$, and put J_g in the hth position of the sequence. If $N_s = \Phi$, then let $s = s-1$; otherwise, go to Step 3.

Step 3. For jobs in T, determine their release times by Algorithm A1, and start processing each job at its release time. For all jobs $J_i \in N_1 \bigcup \cdots \bigcup N_s$, let $x_i = a_i$.

We now establish the running time of Algorithm HA1. In Step 1, the determination of set N_0 requires n operations to check all jobs. In Step 2, if N_0 consists of n_0 jobs, then the selection of each a_g requires at most $(n - n_0 - g + 1)$ operations, so Step 2 requires at most $\frac{(n-n_0+1)(n-n_0)}{2}$ operations. Step 3 requires $(n - n_0)$ operations to determine the release times for the jobs in set T and compress the other jobs. Therefore, Algorithm HA1 has an overall running time of no more than

$\frac{(n-n_0)^2}{2}+\frac{5}{2}n-\frac{3}{2}n_0$, i.e., O($n^2$).

In Algorithm HA1, Step 1 fully compresses all jobs in set N_0 according to Theorem 3. In Step 2, since the determination of the compression status of all jobs in $J\backslash N_0$ is complicated, we only determine some jobs not to be compressed and make the total release time compressing cost as small as possible by utilizing Theorem 2 simultaneously. Step 3 determines some release times by applying Theorem 1.

Using Algorithm HA1, we get the value of the objective function as follows:

$$K_h = v + \sum_{J_i \in J\backslash T} c_i a_i + w[a'_1 + \cdots + ha'_h], \tag{13}$$

where $J'_1, \cdots, J'_h \in T$ and $a'_1 \geq a'_2 \geq \cdots \geq a'_h$.

Denote K^* as the optimal objective value of the given problem, i.e.,

$$K^* = \min_{x \in X, \pi \in \Pi} \{\overline{K}_1^*(x,\pi), \overline{K}_2^*(x,\pi)\}.$$

Algorithm HA1 has the following performance bound.

Theorem 4 $K_h / K^* \leq 2$.

Proof For the case where $p_1 + p_2 + \cdots + p_{k-1} > v$, let $\overline{T}$ denote the set $A_2 \bigcup A_3$ of $\overline{K}_1^*(x,\pi)$, where $\overline{T} = \{\overline{J}_1, \overline{J}_2, \cdots, \overline{J}_t\}$. From Algorithm HA1, $d_m \leq a_m$, and $J_m \notin \overline{T}$, we have $h \leq t$. Denote K as

$$K = v + \sum_{J_i \in J\backslash \overline{T}} c_i a_i + w(\overline{a}_1 + 2\overline{a}_2 + \cdots + t\overline{a}_t), \tag{14}$$

where we assume that the sequence $(\overline{J}_1, \overline{J}_2, \cdots, \overline{J}_t)$ is in decreasing order of their processing times. From (10), we have

$$K + (mw-1)d_m \leq \overline{K}_1^*(x,\pi),$$

that is

$$K \leq \overline{K}_1^*(x,\pi) + (1-mw)d_m.$$

From Algorithm HA1, we have $a'_i \geq \overline{a}_i, i = 1, 2, \cdots, h$. Let $\overline{K}_i$, i=1, 2, …, h, be

the objective function value of the schedule that is obtained by replacing J_i' in the ith position of the schedule, which is generated by Algorithm HA1, by $\overline{J}_i$ in set $\overline{T}$. Without loss of generality, we assume that $T \bigcap \overline{T} = \Phi$, and we have

$$\begin{aligned}\overline{K}_1 &= K_h + c_1'a_1' - \overline{c}_1\overline{a}_1 + w\overline{a}_1 - wa_1' \\ &= K_h + (c_1' - \overline{c}_1)\overline{a}_1 + (c_1' - w)(a_1' - \overline{a}_1) \\ &\geq K_h - (k-1)w\overline{a}_1.\end{aligned}$$

Similarly, for $i = 2, \ldots, h$, we also have

$$\overline{K}_i \geq K_h - (k-1)w\overline{a}_i.$$

Therefore,

$$K \geq K_h - (k-1)w(\overline{a}_1 + \overline{a}_2 + \cdots + \overline{a}_h),$$

that is,

$$K_h \leq K + (k-1)w(\overline{a}_1 + \overline{a}_2 + \cdots + \overline{a}_h).$$

Because $h \leq t$, so

$$\begin{aligned}K_h &\leq K + (k-1)w(\overline{a}_1 + \overline{a}_2 + \cdots + \overline{a}_h) + [\overline{c}_{h+1} - (h+1)w]\overline{a}_{h+1} + \cdots + (\overline{c}_t - tw)\overline{a}_t \\ &\leq \overline{K}_1^*(x, \pi) + (k-1)w(\overline{a}_1 + \overline{a}_2 + \cdots + \overline{a}_t) + (1 - mw)d_m.\end{aligned}$$

From the constraints of problem P1, we have

$$(1 - mw)d_m + (k-1)w(\overline{a}_1 + \overline{a}_2 + \cdots + \overline{a}_t) \leq d_m + (v - d_m) = v \leq \overline{K}_1^*(x, \pi),$$

therefore,

$$K_h \leq 2\overline{K}_1^*(x, \pi),$$

i.e., $$K_h / \overline{K}_1^*(x, \pi) \leq 2.$$

For the case where $p_1 + p_2 + \cdots + p_{k-1} \leq v$, let $\overline{T} = B_2 \bigcup B_3$, and we can similarly prove the result $K_h / \overline{K}_2^*(x, \pi) \leq 2$.

Therefore, $K_h / K^* \leq 2$. ∎

5. Special Cases

Assuming that $c_1 = c_2 = \cdots = c_n = c$ and considering the relation between c and w,

we examine the following special cases:

Case *l*: Assume that $\frac{1}{k} < w \le \frac{1}{k-1}$, $lw \le c < (l+1)w$, and that the sum of the normal processing times of any *l* jobs in *J* is not larger than v. In fact, for different *l*, *l* =1, 2, …, *k* – 1, we have the same conclusions.

In order to reduce the scope to find the optimal schedule for the problem, it is necessary to consider two cases.

i) $p_1 + \cdots + p_{k-1} > v$.

In this case, the objective function $K_1(x, \pi)$ in (4) becomes $\overline{K}_1(x,\pi)$:

$$\begin{aligned}\overline{K}_1(x,\pi) &= \sum_{i=1}^{n}(a_i - x_i) + \sum_{i=1}^{n} cx_i + mwd_m + w[(a_1 - x_1) + 2(a_2 - x_2) \\ &\quad + \cdots + (m-1)(a_{m-1} - x_{m-1})] \\ &= \sum_{i=1}^{m} a_i + \sum_{i=m}^{n}(a_i - x_i) + \sum_{i=m}^{n} cx_i + mwd_m + w[a_1 + \cdots + (m-1)a_{m-1}] \\ &\quad - [(1-c+w)x_1 + \cdots + (1-c+(m-1)w)x_{m-1}].\end{aligned}$$

Thus, the optimal solution for problem P1 is

$$x_1 = 0, \cdots, x_t = 0, x_{t+1} = \bar{a}_{t+1}, x_{t+2} = a_{t+2}, \cdots, x_{m-1} = a_{m-1}, \cdots, x_n = a_n \ ,$$

where $0 \le \bar{a}_{t+1} < a_{t+1}$. The optimal objective value $\overline{K}_1(\pi)$ of problem P1 is

$$\begin{aligned}\overline{K}_1(\pi) &= \sum_{i=1}^{t} a_i + (a_{t+1} - \bar{a}_{t+1}) + mwd_m + c(\bar{a}_{t+1} + \sum_{i=t+2}^{n} a_i) \\ &\quad + w[a_1 + \cdots + ta_t + (t+1)(a_{t+1} - \bar{a}_{t+1})] \\ &= v + (mw-1)d_m + c[\bar{a}_{t+1} + \sum_{i=t+2}^{n} a_i] \\ &\quad + w[(a_1 + \cdots + ta_t + (t+1)(a_{t+1} - \bar{a}_{t+1})]. \qquad (15)\end{aligned}$$

ii) $p_1 + \cdots + p_{k-1} \le v$.

In this case, the objective function $K_2(x, \pi)$ in (5) becomes $\overline{K}_2(x, \pi)$:

$$\begin{aligned}\overline{K}_2(x,\pi) &= v + \sum_{i=k}^{n}(a_i - x_i) + \sum_{i=1}^{n} cx_i + w[(a_1 - x_1) + \cdots + (k-1)(a_{k-1} - x_{k-1})] \\ &= v + \sum_{i=k}^{n} cx_i + w[a_1 + \cdots + (k-1)a_{k-1}] \\ &\quad - [(w-c)x_1 + \cdots + ((k-1)w - c)x_{k-1}].\end{aligned}$$

Thus, the optimal solution for problem P2 is

$$x_1 = 0, \cdots, x_t = 0, x_{t+1} = \bar{a}_{t+1}, x_{t+2} = a_{t+2}, \cdots, x_{k-1} = a_{k-1}, \cdots, x_n = a_n ,$$

where $0 \le \bar{a}_{t+1} < a_{t+1}$. The optimal objective value $\overline{K}_2(\pi)$ of problem P2 is

$$\overline{K}_2(\pi) = v + c(\bar{a}_{t+1} + \sum_{i=t+2}^{n} a_i) + w[a_1 + \cdots + ta_t + (t+1)(a_{t+1} - \bar{a}_{t+1})]. \qquad (16)$$

For both cases i) and ii), when the status of all jobs belonging to set J is determined to be fully compressed or not compressed, the permutation π that satisfies $a_1 \ge a_2 \ge \cdots \ge a_t$ has the smallest objective value.

On the basis of the above discussion, an optimal schedule must be included in all different initial $x \in X$ in problems P1 and P2 with formulations (15) and (16). Next, we develop a new heuristic algorithm for these special cases.

Heuristic Algorithm HA2

Step 1. Sequence the jobs in decreasing order of their normal processing times, i.e., $a'_1 \ge a'_2 \ge \cdots \ge a'_n$.

Step 2. The first l jobs $J'_1, J'_2, \cdots, J'_l$ in the sequence of Step 1 should not be compressed. Sequence them as follows: Let their release times be $r_1 = v - (a'_1 + a'_2 + \cdots a'_l)$, $r_2 = r_1 + a'_1, \cdots,$ $r_l = r_1 + a'_1 + \cdots + a'_{l-1}$, respectively. Furthermore, each job should begin processing at its release time.

Step 3. The other ($n - l$) jobs should be fully compressed.

The running time of Algorithm HA2 can be estimated as follows. Step 1 requires *nlogn* operations to sequence *n* jobs. In Step 2, the determination of the release times of the first *l* jobs requires *l* operations. Step 3 requires ($n - l$) operations to compress ($n - l$) jobs. So, Algorithm HA2 has an overall running time of no more than (n + *nlogn*), i.e., O(*nlogn*).

For Algorithm HA2, it is natural to first sequence the jobs in decreasing order of their normal processing times owing to the peculiarities of the optimal solutions to

problems P1 and P2 for these special cases. Then, Step 2 selects the first l jobs not to be compressed and sequences them by utilizing Theorems 1 and 2.

The objective function value generated by the Algorithm HA2 is

$$K_h' = v + \sum_{i=l+1}^{n} ca_i' + w[a_1' + \cdots + la_l']. \tag{17}$$

Let $a_{\max} = \max\{a_i \mid a_i \in J\}$ and denote K_1* as the optimal objective value of the problem. In the following, we prove that the solution generated by Algorithm HA2 has the following performance bound.

Theorem 5 The solution generated by Algorithm HA2 has a performance bound of

$$K_h' - K_1^* \le (1-c)a_{\max}.$$

Proof First, we prove the conclusion for case i):

Let $a_1' + a_2' + \cdots + a_n' = a_1 + a_2 + \cdots + a_n = M$. From (15), (17) and $t \ge l$, we have

$$\begin{aligned}
\overline{K}_1(\pi) - K_h' &= (mw-1)d_m + c(\bar{a}_{t+1} + \sum_{i=t+2}^{n} a_i) + w[a_1 + \cdots + ta_{t-1} + (t+1)(a_{t+1} - \bar{a}_{t+1})] \\
&\quad - [\sum_{i=l+1}^{n} ca_i' + w(a_1' + \cdots + la_l')] \\
&= (mw-1)d_m + c\{M - [a_1 + \cdots + a_t + (a_{t+1} - \bar{a}_{t+1})]\} \\
&\quad + w[a_1 + \cdots + ta_t + (t+1)(a_{t+1} - \bar{a}_{t+1})] - c[M - (a_1' + \cdots + a_l')] - w(a_1' + \cdots + la_l') \\
&= (mw-1)d_m + [(c-w)a_1' + \cdots + (c-lw)a_l'] \\
&\quad - \{(c-w)a_1 + \cdots + (c-tw)a_t + [c-(t+1)w](a_{t+1} - \bar{a}_{t+1})\} \\
&= (mw-1)d_m + (c-w)(a_1' - a_1) + \cdots + (c-lw)(a_l' - a_l) \\
&\quad - [c-(l+1)w]a_{l+1} - \cdots - (c-tw)a_t - [c-(t+1)w](a_{t+1} - \bar{a}_{t+1}).
\end{aligned}$$

We note that for any integer s, if $s > l$, then $c - sw < 0$; otherwise, $c - sw \ge 0$. Therefore,

$$\overline{K}_1(\pi) - K_h' \ge (mw-1)d_m,$$

and, because $m > l$, we have

$$K'_h - \overline{K}_1(\pi) \le (1-mw)d_m \le (1-c)d_m \le (1-c)a_m \le (1-c)a_{\max},$$

i.e., $$K'_h - \overline{K}_1(\pi) \le (1-c)a_{\max}.$$

For case ii), similar to i), we can obtain the result $K'_h - \overline{K}_2(\pi) \le 0$.

Therefore,

$$K'_h - K_1^* \le (1-c)a_{\max}.$$ أ

It is clear that the performance bound of Algorithm HA2 for the special cases is tighter than that of Algorithm HA 1 for the general problem.

6. Conclusions

In this paper we have considered the single machine scheduling problem with resource dependent release times and processing times, in which both release times and processing times are strictly linear decreasing functions of the amount of resources consumed. Based on an analysis of the optimal properties, we have developed a heuristic algorithm and derived its performance bound. For the special case where all unit processing time compression costs $c_1, c_2, \cdots, c_n$ are equal, we have presented a heuristic algorithm that yields a tighter performance bound.

Further research can be undertaken to develop efficient algorithms for the problem under different parameter constraints such as $b_i \le x_i \le a_i$, i = 1, 2, …, n, and other generalizations.

Acknowledgement

This research was supported in part by the Research Grants Council of Hong Kong under grant number PolyU5245/99H.

References

[1] B. Alidaee and A. Ahmadian, Two parallel machine sequencing problems involving controllable job processing times, *European Journal of Operational*

Research 70 (1993) 335-341.

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

[3] T.C.E. Cheng, Z.L. Chen and C.-L. Li, Parallel-machine scheduling with controllable processing times, *IIE Transactions* 28 (1996) 177-180.

[4] T.C.E. Cheng and A. Janiak, Resource optimal control in some single-machine scheduling problems, *IEEE Transactions on Automatic Control* 39 (1994) 1243-1246.

[5] T.C.E. Cheng, A. Janiak and M.Y. Kovalyov, Single machine batch scheduling with resource dependent setup and processing times, *European Journal of Operational Research* 135 (2001) 177-183.

[6] T.C.E. Cheng and M. Y. Kovalyov, Single machine batch scheduling with deadlines and resource dependent processing times, *Operations Research Letters* 17 (1995) 243-249.

[7] A. Janiak, Time-optimal control in a single machine problem with resource constraints, *Automatica* 22 (1986) 745-747.

[8] A. Janiak, Single machine scheduling problem with a common deadline and resource dependent release dates, *European Journal of Operational Research* 53 (1991)317-325.

[9] A. Janiak and M.Y. Kovalyov, Single machine scheduling subject to deadlines and resource dependent processing times, *European Journal of Operational Research* 94 (1996) 284-291.

[10] C.-L. Li, E.C. Sewell and T.C.E. Cheng, Scheduling to minimize release-time resource consumption and tardiness penalties, *Naval Research Logistics* 42 (1995) 949-966.

[11] E. Nowicki and S. Zdrzalka, A survey of results for sequencing problems with controllable processing times, *Discrete Applied Mathematics* 26 (1990) 271-287.

[12] E. Nowicki and S. Zdrzalka, A bicriterion approach to preemptive scheduling of parallel machines with controllable job processing times, *Discrete Applied Mathematics* 63 (1995) 237-256.

[13]S. S. Panwalkar and R. Rajagopalan, Single-machine sequencing with controllable processing times, *European Journal of Operational Research* 59 (1992) 298-302.

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

[15]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.

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

[17]S. Zdrzalka, Scheduling jobs on a single machine with release dates, delivery times and controllable processing times: worst-case analysis, *Operations Research Letters* 10 (1991) 519-523.

[18]F. Zhang, G. Tang and Z. L. Chen, A 3/2-approximation algorithm for parallel machine scheduling with controllable processing times, *Operations Research Letters* 29 (2001) 41-47.