Scheduling a batch processing machine subject to precedence constraints, release dates and identical processing times

T.C.E. Cheng[1*], **J.J. Yuan**[1,2] **and A.F. Yang**[1,2]

[1]Department of Management, The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong, People's Republic of China

[2]Department of Mathematics, Zhengzhou University,
Zhengzhou, Henan 450052, People's Republic of China

ABSTRACT

We consider the single machine parallel-batching scheduling problem with precedence relations, release dates and identical processing times to minimize a regular objective function. When the processing times are unit, we give an $O(n^2)$ time optimal algorithm. When there are no precedence relations, we solve this problem by dynamic programming in $O(n^3)$ time. When the precedence relations are "layerly complete", we solve this problem by a dynamic programming algorithm that runs in $O(n^7)$ time. For the total weighted completion time minimization problem, we give an $O(n^2)$ time $\frac{3}{2}$-approximation algorithm. For the makespan minimization problem, we give an $O(n^2)$ time optimal algorithm and an expression of the minimum makespan.

Keywords: Scheduling, Batch processing, Precedence constraints, Release dates, Identical processing times, Algorithms,.

1 Introduction and Problem Formulation

Let n jobs $J_1, J_2, ..., J_n$ and a single machine that can handle batch jobs at the same time be given. There are precedence relations $\prec$ between the jobs. Each job J_j has an integer

*Corresponding author

processing time p_j and an integer release date r_j. The jobs are processed in batches, where a batch is a subset of jobs and we require that the batches form a partition of the set of all jobs. The processing time of a batch is equal to the longest processing time of all the jobs in the batch.

Suppose that a batch sequence (which indicates the processing order of a certain batch partition of the jobs) is given and we will process the batched jobs according to this batch sequence. We require that the starting time of a batch is at least the maximum release date of the jobs in it, and this maximum value can be regarded as the release date of the batch. If J_i and J_j are two jobs such that $J_i \prec J_j$, we also require that J_j be processed after the completion time of J_i; so J_i and J_j cannot be processed in the same batch. The completion time of all the jobs in a batch is defined as the completion time of the batch. Following [1] and [7], we call this model the parallel-batching scheduling problem and denote it by

$$1|\textit{prec; p-batch; } r_j|f,$$

where f is the objective function, which is a function of the job completion times C_j under a given schedule, to be minimized. In this paper we will suppose that the objective function f is regular [1], i.e., f is nondecreasing in C_j.

For the problem $1|\textit{prec; p-batch; } r_j|f$, a feasible schedule is given by a batch sequence

$$BS = (B_1, B_2, ..., B_N)$$

such that, for any pair of jobs J_i and J_j with $J_i \prec J_j$, if $J_i \in B_x$ and $J_j \in B_y$, then $x < y$. Since the objective function is regular, we suppose that all the jobs in the same batch start simultaneously at the earliest possible starting time. Consequently, the starting time of each batch is determined by the batch sequence. For each batch B_x, if the starting time of B_x is s_x, then the completion time of B_x is simply

$$s_x + \max_{J_j \in B_x} p_j.$$

The parallel-batching scheduling problem is one of the important modern scheduling models that has received much attention in the literature. The fundamental model of the parallel-batching scheduling problem was first introduced by Lee *et al.* in [8] with the restriction that the number of jobs in each batch is bounded by a number b, which is denoted by $1|\textit{p-batch}; b < n|f$. This bounded model is motivated by the *burn-in* operations in semiconductor manufacturing [8]. For example, a batch of integrated circuits (jobs) are put inside an oven of limited size to test for their thermal standing ability. The circuits are heated inside the oven until all circuits are burned. The burn-in time of the circuits (job processing times) may be different. When a circuit is burned, it has to wait inside the oven until all circuits are burned. Therefore, the processing time of a batch of circuits is the processing time of the longest job in the batch.

An extensive discussion of the unbounded version of the problem under study is provided in [2]. This unbounded model can be applied, for example, to situations where the

batch contents need to be hardened in a sufficiently large kiln and so the batch size is not restricted [2].

Recent developments of this topic can be found in the book [1] and the web site [3]. In addition, [4], [6], [9] and [10] presented new complexity results on the parallel-batching scheduling problem subject to release dates. We will only consider the unbounded version of the parallel-batching scheduling problem.

For the problem $1|prec;\ p_j = p;\ p\text{-}batch|f$, it is implied in [1] that there is an $O(n^2)$ time algorithm for every regular objective function. For the problem $1|prec;\ p\text{-}batch|f$, Cheng *et al.* [5] recently showed that even the simplest problems $1|chains;\ p\text{-}batch|C_{\max}$ and $1|chains;\ p\text{-}batch|\sum C_j$ are strongly NP-hard.

We consider the problem $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|f$. It is reported in [3] that the complexity of the problem $1|prec;\ p_j = 1;\ p\text{-}batch;\ r_j|f$ is unknown even for such common regular objective functions as makespan, $C_{\max} = \max\{C_j\}$; maximum lateness, $L_{\max} = \max\{C_j - d_j\}$; total completion time, $\sum C_j$; and total tardiness, $\sum T_j$, where d_j is the due date of J_j and $T_j = \max\{0, C_j - d_j\}$.

We show in this paper that the scheduling problem $1|prec;\ p_j = 1;\ p\text{-}batch;\ r_j|f$ can be solved in $O(n^2)$ time, $1|p_j = p;\ p\text{-}batch;\ r_j|f$ can be solved in $O(n^3)$ time, and $1|complete\text{-}prec;\ p_j = p;\ p\text{-}batch;\ r_j|f$ can be solved in $O(n^7)$ time. We give an $O(n^2)$ time $\frac{3}{2}$-approximation algorithm for the problem $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|\sum w_jC_j$. We also show that the problem $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|C_{\max}$ can be solved in $O(n^2)$ time. Furthermore, we give an expression of the minimum makespan.

2 Release Date Modification

Let us consider the problem $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|f$, where f is any regular objective function. This problem can be solved by a simple algorithm.

We suppose that the job enumeration given here is topological, i.e., for any two jobs J_i and J_j with $J_i \prec J_j$, we must have $i < j$. According to Brucker [1], a topological job enumeration can be obtained in $O(n^2)$ time by the standard "Algorithm Topological Enumeration".

If J_i and J_j are two jobs such that $J_i \prec J_j$, then the starting time of the batch that contains J_j must be at least $r_i + p$ in any feasible schedule. Modifying the release date of each job J_j by setting

$$r'_j := \max\{r_j, r_i + p\},$$

we see that the value of the objective function will not change under any feasible schedule. Hence, we can recursively modify the release dates of the jobs such that, for each pair of jobs J_i and J_j, if $J_i \prec J_j$, then $r'_i + p \le r'_j$. This procedure can be done in $O(n^2)$ time by the standard "Algorithm Modify r_j" in Brucker [1].

An important observation is that, under the modified release dates, $r'_j \ge r'_i + p$ if

$J_i \prec J_j$. Furthermore, in any feasible schedule, the starting time of the batch that contains J_j is at least r'_j. This greatly simplifies the description of the algorithms discussed in this paper. Hence, we assume in the rest of this paper that the release dates have initially been modified such that for each pair of jobs J_i and J_j, if $J_i \prec J_j$, then $r_i + p \leq r_j$.

Consider the special case where there is a certain integer e with $0 \leq e \leq p-1$ such that, for each r_j, there exists k_j such that $r_j = e + k_j p$, $1 \leq j \leq n$. Suppose we have k distinct release dates $r^{(1)}, r^{(2)}, ..., r^{(k)}$ such that $r^{(i)} < r^{(i+1)}$ for $1 \leq i \leq k-1$. By noting that $r^{(i)} < r^{(i+1)}$ implies that $r^{(i)} + p \leq r^{(i+1)}$, we can form a batch sequence

$$BS = (B_1, B_2, ..., B_k)$$

in $O(n)$ time by setting

$$B_x = \{J_j : r_j = r^{(x)}\}, \text{ for } 1 \leq x \leq k.$$

Clearly, $BS = (B_1, B_2, ..., B_k)$ is a feasible schedule such that the starting time and the completion time of each batch B_x are $r^{(x)}$ and $r^{(x)} + p$, respectively. Since any job J_j starts at its release date (i.e., the earliest possible starting time) r_j, the batch sequence BS must be optimal for any regular objective function f.

Based on the above discussion, the problem $1|$*prec;* $p_j = p$*; p-batch;* $r_j = e + k_j p|f$ can be solved by the following batching rule.

Algorithm 2.1 Batching Rule for Job System with $r_j = e + k_j p$.

At each point, form the next first batch by including all available unbatched jobs that have no unbatched predecessors.

When $p_j = 1$ for all jobs J_j, $e = 0$, and so the above batching rule solves the problem $1|$*prec;* $p_j = 1$*; p-batch;* $r_j|f$.

3 Job System under Identical Processing Times

Define

$$\mathcal{T} = \{r_j + xp : 1 \leq j \leq n, 0 \leq x \leq n\}.$$

It is clear that there must be an optimal schedule for $1|$*prec;* $p_j = p$*; p-batch;* $r_j|f$ such that the starting time and the completion time of every batch belong to $\mathcal{T}$. Hence, it suffices to consider the schedule with batch starting times in $\mathcal{T}$.

Given an instance of the problem $1|$*prec;* $p_j = p$*; p-batch;* $r_j|f$, let $\mathcal{J} = \{J_1, J_2, ..., J_n\}$. We define layers of the jobs in the following way:

$$\mathcal{L}_1 = \{J_j : J_j \text{ has no predecessors in } \mathcal{J}\},$$

$$\mathcal{L}_i = \{J_j : J_j \text{ has no predecessors in } \mathcal{J} \setminus (\cup_{1 \leq j \leq i-1} \mathcal{L}_j)\}, \ \ 2 \leq i \leq n.$$

$\mathcal{L}_i$ is called the i-th layer of the job system. Let $m = \max\{i : \mathcal{L}_i \text{ is not empty}\}$. Then $\cup_{1\le i\le m}\mathcal{L}_i = \mathcal{J}$ and each layer $\mathcal{L}_i$ consists of independent jobs. Clearly, the layers of jobs can be obtained in $O(n^2)$ time.

If, for $1 \le i \le m-1$, each job in $\mathcal{L}_i$ is a predecessor of every job in $\mathcal{L}_{i+1}$, the precedence relation is said to be "layerly complete". The corresponding problem is denoted by $1|\textit{complete-prec; } p_j = p\textit{; p-batch; } r_j|f$.

3.1 Independent Job System

A special subproblem of $1|\textit{prec; } p_j = p\textit{; p-batch; } r_j|f$ is $1|p_j = p\textit{; p-batch; } r_j|f$, where the jobs are independent (with no precedence constraints between jobs). We first consider the problem

$$\mathcal{P}_f(\mathcal{L}, s, t): \quad 1|p_j = p\textit{; p-batch; } r_j\textit{; } J_j \in \mathcal{L}|f$$

under the restriction that the starting time of the first batch is at least s and the completion time of the last batch is at most t, where either $f = \sum_{J_j\in\mathcal{L}} f_j$ or $f = \max_{J_j\in\mathcal{L}} f_j$.

Let $\mathrm{OPT}_f(\mathcal{L}, s, t)$ denote the optimal (minimum) objective value for $\mathcal{P}_f(\mathcal{L}, s, t)$. We take the convention that $\mathrm{OPT}_f(\mathcal{L}, s, t) = 0$ if $\mathcal{L} = \emptyset$, and $\mathrm{OPT}_f(\mathcal{L}, s, t) = +\infty$ if $\mathcal{P}_f(\mathcal{L}, s, t)$ has no feasible schedule. Let

$$R(\mathcal{L}, s, t) = \{s\} \cup \{r_j : J_j \in \mathcal{L}, r_j + p \le t\}.$$

Then $|R(\mathcal{L}, s, t)| \le |\mathcal{L}|$. It is clear that there is an optimal schedule for $\mathcal{P}_f(\mathcal{L}, s, t)$ such that the starting time of the first batch belongs to $R(\mathcal{L}, s, t)$. Suppose that the starting time of the first batch is r, and let

$$\mathcal{L}^{(r)} = \{J_i \in \mathcal{L} : r_j \le r\}.$$

Since f is regular, $\mathcal{L}^{(r)}$ will form the first batch. Define $g(\mathcal{L}^{(r)})$ by

$$g(\mathcal{L}^{(r)}) = \begin{cases} \sum_{J_j\in\mathcal{L}^{(r)}} f_j(r+p), & \text{if } f = \sum f_j, \\ \max_{J_j\in\mathcal{L}^{(r)}} f_j(r+p), & \text{if } f = \max f_j. \end{cases}$$

Clearly, the computing of $g(\mathcal{L}^{(r)})$ needs at most $O(|\mathcal{L}|)$ time. For $f = \sum f_j$, we have the following dynamic programming recursion

$$\begin{aligned} & \mathrm{OPT}_f(\mathcal{L}, s, t) \\ = \; & \min_{r\in R(\mathcal{L},s,t)} \left(g(\mathcal{L}^{(r)}) + \mathrm{OPT}_f(\mathcal{L} \setminus \mathcal{L}^{(r)}, r+p, t) \right). \end{aligned}$$

For $f = \max f_j$, we have

$$\begin{aligned} & \mathrm{OPT}_f(\mathcal{L}, s, t) \\ = \; & \min_{r\in R(\mathcal{L},s,t)} \max \left\{ g(\mathcal{L}^{(r)}), \mathrm{OPT}_f(\mathcal{L} \setminus \mathcal{L}^{(r)}, r+p, t) \right\}. \end{aligned}$$

In the dynamic programming procedure, the lower bounds of the starting times of the job subsets of $\mathcal{L}$ are chosen from

$$\{s + xp \in \mathcal{T} : 1 \le x \le |\mathcal{L}|\} \cup \{r_j + xP : J_j \in \mathcal{L},\ 1 \le x \le |\mathcal{L}_i|\},$$

and we have $O(|\mathcal{L}|^2)$ such choices. When a new lower bound r of the starting time is established, the new job subset $\mathcal{L} \setminus \mathcal{L}^{(r)}$ is uniquely formed. Each recursion runs in $O(|\mathcal{L}|)$ time. Hence, when s and t are given, both dynamic programming recursions run in $O(|\mathcal{L}|^3)$ time to solve the problem $\mathcal{P}_f(\mathcal{L}, s, t)$.

For the general problem $1|p_j = p;\ p\text{-}batch;\ r_j|f$, the set of jobs is $\mathcal{L} = \mathcal{J}$. Since

$$\mathrm{OPT}_f = \mathrm{OPT}_f(\mathcal{J}, 0, \max_{1 \le j \le n} r_j + np),$$

we conclude that the problem $1|p_j = p;\ p\text{-}batch;\ r_j|f$ can be solved in $O(n^3)$ time.

3.2 Layerly Complete Precedence Contrained Job System

Now we turn our attention to the problem $1|complete\text{-}prec;\ p_j = p;\ p\text{-}batch;\ r_j|f$. Let $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_m$ be the layers of jobs under $\prec$. Since each job in $\mathcal{L}_i$ must complete its processing before the starting of any job in $\mathcal{L}_{i+1}$, this enables us to use dynamic programming to solve this problem. Let

$$\mathcal{L}_i^* = \mathcal{L}_i \cup \mathcal{L}_{i+1} \cup ... \cup \mathcal{L}_m,\ \ 1 \le i \le m.$$

For $s \in \mathcal{T} = \{r_j + xp : 1 \le j \le n, 1 \le x \le n\}$, we consider the problem

$$\mathcal{P}_f(\mathcal{L}_i^*, s): \quad 1|complete\text{-}prec;\ p_j = p;\ p\text{-}batch;\ r_j; J_j \in \mathcal{L}_i^*|f$$

under the restriction that the starting time of the first batch is at least s. Let $\mathrm{OPT}_f(\mathcal{L}_i^*, s)$ denote the optimal (minimum) objective value for $\mathcal{P}_f(\mathcal{L}_i^*, s)$. We take the convention that $L_{m+1}^* = \emptyset$ and $\mathrm{OPT}_f(\emptyset, s) = 0$. Let

$$R(i, s) = \{s + xp \in \mathcal{T} : 1 \le x \le |\mathcal{L}_i|\} \cup \{r_j + xp : J_j \in \mathcal{L}_i,\ 1 \le x \le |\mathcal{L}_i|\}.$$

Then, we have the dynamic programming recursions

$$\mathrm{OPT}_f(\mathcal{L}_i^*, s) = \min_{t \in R(i,s)} \Big(\mathrm{OPT}_f(\mathcal{L}_i, s, t) + \mathrm{OPT}_f(\mathcal{L}_{i+1}^*, t)\Big)$$

for $f = \sum f_j$, and

$$\mathrm{OPT}_f(\mathcal{L}_i^*, s) = \min_{t \in R(i,s)} \max\Big\{\mathrm{OPT}_f(\mathcal{L}_i, s, t), \mathrm{OPT}_f(\mathcal{L}_{i+1}^*, t)\Big\}$$

for $f = \max f_j$. Now we have m choices for i, $O(n^2)$ choices for $s \in \mathcal{T}$ and $O(n^2)$ choices for $t \in R(i, s)$. Before the recursion, computing all $\mathrm{OPT}_f(\mathcal{L}_i, s, t)$ for all i, s and t needs $O(n^4 \sum_{1 \le i \le m} |\mathcal{L}_i|^3) = O(n^7)$ time. Hence, the complexity of each of the two dynamic programming recursions is $O(mn^4 + n^7) = O(n^7)$. Since the optimal value of the problem $1|complete\text{-}prec;\ p_j = p;\ p\text{-}batch;\ r_j|f$ is given by $\mathrm{OPT}_f = \mathrm{OPT}_f(\mathcal{L}_1^*, 0)$, we conclude that the scheduling problem $1|complete\text{-}prec;\ p_j = p;\ p\text{-}batch;\ r_j|f$ can be solved in $O(n^7)$ time.

3.3 An Approximation Algorithm

Although the complexity of $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|f$ remains open, we do not expect a simple polynomial-time algorithm for this problem. In the following, we will give a release date rounding polynomial-time approximation algorithm.

Denote each release date r_j by

$$r_j = e_j + k_j p,\ \ 0 \le e_j \le p-1 \text{ and } k_j = \lfloor \frac{r_j}{p} \rfloor.$$

Let x be an integer with $0 \le x \le p-1$. The release dates are rounded in the following way:

$$r_j^*(x) = \begin{cases} x + k_j p, & \text{if } e_j \le x, \\ x + (k_j + 1)p, & \text{if } e_j > x. \end{cases}$$

An important observation is that $0 \le r_j^*(x) - r_j \le p-1$, for $1 \le j \le n$.

Lemma 3.3.1 If $J_i \prec J_j$, then $r_i^*(x) + p \le r_j^*(x)$.

Proof Since $J_i \prec J_j$, we have $r_i + p \le r_j$. For the reason that $r_i^*(x) - r_i \le p-1$ and $0 \le r_j^*(x) - r_j$, we deduce $r_i^*(x) + p \le r_j^*(x) + p - 1$, and thus $r_i^*(x) < r_j^*(x)$. The result follows by noting that $r_j^*(x) - r_i^*(x)$ can be divided by p.

□

Under the release dates $r_j^*(x)$, $1 \le j \le n$, the problem $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j^*(x)|f$ can be solved by Algorithm 2.1 to obtain an optimal schedule $\pi(x)$. This schedule $\pi(x)$ is clearly feasible for $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|f$, and so can be used as an approximation solution.

Since the starting time of each job J_j under $\pi(x)$ is exactly $r_j^*(x)$, we have $C_j(\pi(x)) = r_j^*(x) + p$, $1 \le j \le n$. So, the objective value is given by

$$F(x) = f(r_1^*(x) + p, r_2^*(x) + p, ..., r_n^*(x) + p).$$

The following lemma shows that, to minimize $F(x)$ for $0 \le x \le p-1$, we only need to minimize $F(e_j)$ for $1 \le j \le n$

Lemma 3.3.2 There is an integer j with $1 \le j \le n$ such that

$$F(e_j) = \min\{F(x) : 0 \le x \le p-1\}.$$

Proof Suppose that $0 \le x^* \le p-1$ such that $F(x^*) = \min\{F(x) : 0 \le x \le p-1\}$. If $x^* \notin \{e_j : 1 \le j \le n\}$, we define

$$e^* = \begin{cases} \max\{e_j : 1 \le j \le n\}, & \text{if } \{e_j : e_j < x^*, 1 \le j \le n\} = \emptyset, \\ \max\{e_j : e_j < x^*, 1 \le j \le n\}, & \text{otherwise.} \end{cases}$$

It can be verified that $r_j^*(e^*) < r_j^*(x^*)$, and so, $C_j(\pi(e^*) < C_j(\pi(x^*))$, $1 \leq j \leq n$. Since $f(C_1, C_2, ..., C_n)$ is regular, we must have $F(e^*) \leq F(x^*)$. The result follows.

□

Now, our release date rounding algorithm can be summarized as follows.

Algorithm 3.3.3 Release Date Rounding.

(1) Pick $e^* \in \{e_j : 1 \leq j \leq n\}$ such that

$$F(e^*) = \min\{F(e_j) : 1 \leq j \leq n\}.$$

(2) Apply Algorithm 2.1 to the scheduling problem $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j^*(e^*)|f$ to obtain a schedule $\pi(e^*)$.

Algorithm 3.3.3 is polynomial and has good performance for $f = L_{\max}$ and $f = \sum w_j C_j$. For $f = L_{\max}$, the absolute error $F(e^*) - \mathrm{OPT}_f$ can be bounded from above by $p - 1$, since $r_j^*(e^*) \leq r_j + (p-1)$ for $1 \leq j \leq n$. In the following, we will estimate an upper bound of the performance ratio of Algorithm 3.3.3 for $f = \sum w_j C_j$.

Theorem 3.3.4 Algorithm 3.3.3 is a polynomial-time $\frac{3}{2}$-approximation algorithm.

Proof Let π' be an optimal schedule for $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|\sum w_j C_j$. Then, $C_j(\pi') \geq r_j + p$, $1 \leq j \leq n$. Since $C_j(\pi(e^*)) = r_j^*(e^*) + p$, $1 \leq j \leq n$, we have

Claim 1 $\sum_{1 \leq j \leq n} w_j C_j(\pi(e^*)) - \sum_{1 \leq j \leq n} w_j C_j(\pi') \leq \sum_{1 \leq j \leq n} w_j (r_j^*(e^*) - r_j)$.

Suppose that $|\{e_j : 1 \leq j \leq n\}| = k$ and let

$$\{e_j : 1 \leq j \leq n\} = \{e^{(1)}, e^{(2)}, ..., e^{(k)}\}$$

be such that $e^{(1)} < e^{(2)} < ... < e^{(k)}$. Write $w^{(i)} = \sum_{j:\, e_j = e^{(i)},\ 1 \leq j \leq n} w_j$.

Since $F(e) = \sum_{1 \leq j \leq n} w_j (r_j^*(e) + p_j)$ and $F(e^*) = \min_{1 \leq i \leq k} F(e^{(i)})$, we further have $\sum_{1 \leq j \leq n} w_j r_j^*(e^*) \leq \sum_{1 \leq j \leq n} w_j r_j^*(e^{(i)}$ for $1 \leq i \leq k$. This implies

Claim 2 $\sum_{1 \leq j \leq n} w_j (r_j^*(e^*) - r_j) \leq \sum_{1 \leq j \leq n} w_j (r_j^*(e^{(i)} - r_j)$ for $1 \leq i \leq k$.

By the definition of $r_j^*(e)$, for $1 \leq i \leq k$ and $1 \leq j \leq n$, we have

$$r_j^*(e^{(i)}) - r_j = e^{(i)} - e_j, \text{ if } e_j \leq e^{(i)}$$

and

$$r_j^*(e^{(i)}) - r_j = e^{(i)} - e_j + p, \text{ if } e_j > e^{(i)}.$$

Hence, for $1 \leq i \leq k$, we have

$$\begin{aligned} & \sum_{1 \leq j \leq n} w_j (r_j^*(e^{(i)}) - r_j) \\ = \ & \sum_{j:1 \leq j \leq n,\ e_j \leq e^{(i)}} w_j (e^{(i)} - e_j) + \sum_{j:1 \leq j \leq n,\ e_j > e^{(i)}} w_j (e^{(i)} - e_j + p) \\ = \ & \sum_{1 \leq j \leq n} w_j (e^{(i)} - e_j) + \sum_{j:1 \leq j \leq n,\ e_j > e^{(i)}} w_j p. \end{aligned}$$

This can be rewritten as

Claim 3 $\sum_{1\le j\le n} w_j(r_j^*(e^{(i)}) - r_j) = \sum_{1\le x\le k} w^{(x)}(e^{(i)} - e^{(x)}) + \sum_{i+1\le x\le k} w^{(x)}p$, for $1 \le i \le k$.

Let $Q(i) = \sum_{1\le x\le k} w^{(x)}(e^{(i)} - e^{(x)}) + \sum_{i+1\le x\le k} w^{(x)}p$, for $1 \le i \le k$. Then, by Claim 2 and Claim 3, we have

Claim 4 $\sum_{1\le j\le n} w_j(r_j^*(e^*) - r_j) \le \left(\sum_{1\le i\le k} w^{(i)}\right)^{-1} \sum_{1\le i\le k} w^{(i)}Q(i)$.

By noting the fact that

$$\sum_{1\le i\le k} w^{(i)} \sum_{1\le x\le k} w^{(x)}(e^{(i)} - e^{(x)}) = 0,$$

we can easily deduce that

$$\sum_{1\le i\le k} w^{(i)}Q(i) = \sum_{1\le i\le k} w^{(i)} \sum_{i+1\le x\le k} w^{(x)}p.$$

Equivalently, we have

Claim 5 $\sum_{1\le i\le k} w^{(i)}Q(i) = p \cdot \sum_{1\le i<x\le k} w^{(i)}w^{(x)}$.

Since

$$\sum_{1\le i<x\le k} w^{(i)}w^{(x)} < \frac{1}{2}\left(\sum_{1\le i\le k} w^{(i)}\right)^2,$$

We deduce from Claim 4 and Claim 5 that

$$\begin{aligned}
& \textstyle\sum_{1\le j\le n} w_j(r_j^*(e^*) - r_j) \\
\le\ & \left(\textstyle\sum_{1\le i\le k} w^{(i)}\right)^{-1} \cdot p \cdot \textstyle\sum_{1\le i<x\le k} w^{(i)}w^{(x)} \\
<\ & \tfrac{1}{2} \cdot p \cdot \textstyle\sum_{1\le i\le k} w^{(i)} \\
=\ & \tfrac{1}{2} \textstyle\sum_{1\le j\le n} w_j p \\
\le\ & \tfrac{1}{2} \textstyle\sum_{1\le j\le n} w_j C_j(\pi').
\end{aligned}$$

It follows from Claim 1 that

$$\begin{aligned}
& \textstyle\sum_{1\le j\le n} w_j C_j(\pi(e^*)) \\
\le\ & \textstyle\sum_{1\le j\le n} w_j C_j(\pi') + \sum_{1\le j\le n} w_j(r_j^*(e^*) - r_j) \\
<\ & \tfrac{3}{2} \textstyle\sum_{1\le j\le n} w_j C_j(\pi').
\end{aligned}$$

The result follows.

□

We conjecture that the bound $\frac{3}{2}$ in Theorem 3.4 can be further improved.

4 Makespan Minimization and an Extension

4.1 Makespan Minimization

The problem $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|C_{\max}$, denoted by $\mathcal{P}$ in the sequel, can easily be solved by the following algorithm.

Algorithm 4.1.1 Makespan Minimization Batching Rule.

At each point, form the next last batch by including all unbatched jobs that have no unbatched successors.

Clearly, the complexity of Algorithm 4.1 is $O(n^2)$. The correctness of Algorithm 4.1 is implied by the following theorem.

Theorem 4.1.2 The makespan of $\mathcal{P}$ obtained by Algorithm 4.1 is $p + \max_{1\leq j\leq n} r_j$.

Proof Suppose that $BS = (B_1, B_2, ..., B_k)$ is the batch sequence obtained by Algorithm 4.1. Let $r^{(i)} = \max\{r_j : J_j \in B_i\}$. Then, $r^{(k)} = \max_{1\leq j\leq n} r_j$. For every pair of adjacent batches B_i and B_{i+1}, $1 \leq i \leq k-1$, let $J_x \in B_i$ be such that $r_x = r^{(i)}$. By the batching rule of Algorithm 4.1, there must be a certain $J_y \in B_{i+1}$ such that $J_x \prec J_y$. Then $r_x + p \leq r_y$. This means that $r^{(i)} + p \leq r^{(i+1)}$, for $1 \leq i \leq k-1$. Hence, under the batch sequence BS, each batch B_i has starting time $r^{(i)}$ and completion time $r^{(i)} + p$. The result follows.

□

4.2 An Extention

We consider an extension of problem $\mathcal{P}$. In practice, firms seek to reduce the stocking cost $\sum_{1\leq j\leq n} w_j(C_{\max} - C_j)$ for the ordered goods. So, they face a primary-secondary criterion scheduling problem. The first criterion is to minimize $C_{\max} = \max_{1\leq j\leq n} C_j$, and the second criterion is to maximize $\sum_{1\leq j\leq n} w_j C_j$. We denote this primary-secondary criterion problem by $\mathcal{P}^*$.

Let $BS = (B_1, B_2, ..., B_k)$ be the batch sequence obtained by Algorithm 4.1. Let $t = p + \max_{1\leq j\leq n} r_j$ be the minimum makespan of $\mathcal{P}$. By the construction of BS, for every job $J^{(i)} \in B_i$, there must be a chain of jobs

$$J^{(i)} \prec J^{(i+1)} \prec ... \prec J^{(k)}$$

such that $J^{(x)} \in B_x$ for $i + 1 \leq x \leq k$. It follows that each job in B_i has a completion time of at most $t - (k-i)p$ and a starting time of at most $t - (k-i+1)p$ in any feasible schedule. To maximize $\sum_{1\leq j\leq n} w_j C_j$, we define BS^* as the schedule obtained from BS such that each batch B_i has a starting time of $t - (k-i+1)p$ and a completion time of $t - (k-i)p$. It is clear that BS^* is a feasible schedule and each job completes at its

largest possible completion time under the restriction that the makespan is minimized. Hence, we conclude the following theorem.

Theorem 4.2.1 BS^* is an optimal schedule for the primary-secondary criterion problem $\mathcal{P}^*$.

5 Conclusion

The parallel-batching scheduling problem $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|f$ was studied in this paper. The restriction on jobs with identical processing times largely simplifies the problem, but the presence of the precedence constraints between jobs increases the hardness of the problem. We showed in this paper that the problem $1|prec;\ p_j = 1;\ p\text{-}batch;\ r_j|f$ can be solved in $O(n^2)$ time, $1|p_j = p;\ p\text{-}batch;\ r_j|f$ can be solved in $O(n^3)$ time, and $1|complete\text{-}prec;\ p_j = p;\ p\text{-}batch;\ r_j|f$ can be solved in $O(n^7)$ time. We gave an $O(n^2)$ time $\frac{3}{2}$-approximation algorithm for the problem $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|\sum w_jC_j$. We also showed that the problem $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|C_{\max}$ can be solved in $O(n^2)$ time. Furthermore, we gave an expression of the minimum makespan. For further work, the complexity of $1|prec;\ p_j = p;\ p\text{-}batch;\ r_j|f$ is still open in general for the regular objective function

$$f \in \{L_{\max},\ \sum C_j,\ \sum w_jC_j,\ \sum U_j,\ \sum w_jU_j,\ \sum T_j,\ \sum w_jT_j\}.$$

Acknowledgements

We are grateful for the constructive comments of the referees on an earlier version of this paper. This research was supported in part by The Hong Kong Polytechnic University under a grant from the ASD in *China Business Services*. The last two authors were also supported in part by the National Natural Science Foundation of China.

References

[1] P. Brucker, *Scheduling Algorithms*, Springer-Verlag, Berlin, 2001.

[2] P. Brucker, A. Gladky, H. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn and S.L. van de Velde, Scheduling a batching machine, *Journal of Scheduling*, **1**(1998), 31-54.

[3] P. Brucker and S. Knust, Complexity results for scheduling problems, *http://www.mathematik.uni-osnabrueck.de/research/OR/class/*, 2003.

[4] T.C.E. Cheng, Z.H. Liu and W.C. Yu, Scheduling jobs with release dates and deadlines on a batch processing machine, *IIE Transactions*, **33**(2001), 685-690.

[5] T.C.E. Cheng, C.T. Ng, J.J. Yuan and Z.H. Liu, Scheduling a batch processing machine subject to precedence constraints (*in submission*).

[6] X. Deng and Y.Z. Zhang, Minimizing mean response time for batch processing systems, *Lecture notes in Computer Science*, **1627**(1999), 231-240.

[7] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problems, *Annals of Discrete Mathematics,* **1**(1977), 343-362.

[8] C.-Y. Lee, R. Uzsoy and L.A. Martin-Vega, Efficient algorithms for scheduling semiconductor burn-in operations, *Operations Research*, **40**(1992), 764-775.

[9] Z.H. Liu and W.C. Yu, Scheduling one batch processor subject to job release dates, *Discrete Applied Mathematics*, **105**(2000), 129-136.

[10] Z.H. Liu, J.J. Yuan and T.C.E. Cheng, On scheduling an unbounded batch machine, *Operation Research Letters*, **31**(2003), 42-48.