Competitive two-agent scheduling with deteriorating jobs on a single parallel-batching machine

Lixin Tang[a,*], Xiaoli Zhao[a], Jiyin Liu[b], Joseph Y.-T. Leung[c]

[a] *Liaoning Key Laboratory of Manufacturing System and Logistics, Institute of Industrial Engineering and Logistics Optimization, Northeastern University, Shenyang, 110819, China*
[b] *School of Business and Economics, Loughborough University, Leicestershire LE11 3TU, UK*
[c] *School of Management, Hefei University of Technology, Hefei, Anhui, 230009, P. R. China and Department of Computer Science, New Jersey Institute of Technology, Newark, New Jersey 07102, USA*

Abstract

We consider a scheduling problem in which the jobs are generated by two agents and have time-dependent proportional-linear deteriorating processing times. The two agents compete for a common single batching machine to process their jobs, and each agent has its own criterion to optimize. The jobs may have identical or different release dates. The batching machine can process several jobs simultaneously as a batch and the processing time of a batch is equal to the longest of the job processing times in the batch. The problem is to determine a schedule for processing the jobs such that the objective of one agent is minimized, while the objective of the other agent is maintained under a fixed value. For the unbounded model, we consider various combinations of regular objectives on the basis of the compatibility of the two agents. For the bounded model, we consider two different objectives for incompatible and compatible agents: minimizing the makespan of one agent subject to an upper bound on the makespan of the other agent and minimizing the number of tardy jobs of one agent subject to an upper bound on the number of tardy jobs of the other agent. We analyze the computational complexity of various problems by either demonstrating that the problem is intractable or providing an efficient exact algorithm for the problem. Moreover, for certain problems that are shown to be intractable, we provide efficient algorithms for certain special cases.

Keywords: Scheduling, Release dates, Agent scheduling, Deterioration, Parallel-batching

*Corresponding author. Tel./fax: +86 24 83680169.
E-mail addresses: lixintang@mail.neu.edu.cn (L. Tang)

1. Introduction

This paper addresses a two-agent scheduling problem on a single batching machine with time-dependent proportional-linear deteriorating job processing times. Each agent has a set of jobs to be scheduled on a common batching machine and seeks to minimize a cost function that only depends on the completion times of its own jobs. The machine can process several jobs simultaneously as a batch. The processing time of a batch is the longest processing time of the jobs assigned to the batch. The processing time of a job is a proportional-linear increasing function of its starting time. The problem we are considering is to find a schedule that minimizes the objective of one agent with the restriction that the objective of the other agent cannot exceed a given bound.

Our study is motivated by a production scheduling problem for the ingot soaking process of a primary rolling plant in the steel industry (see Figure 1). When the molten steel from a steelmaking furnace is ready, it is first cast into ingots. This casting process includes pouring the molten steel into molds, followed by solidifying and stripping to remove the molds. The ingots are then reheated and soaked in a soaking pit to the required rolling temperature before being rolled into steel products, which may be either sold directly to customers or used for further processing in the hot rolling mill. The sales department and the hot rolling mill can be viewed as two agents for the ingot soaking process. An ingot can be viewed as a job. The soaking pit can process several jobs simultaneously and can be considered as a bounded parallel-batching machine. The time required for reheating and soaking an ingot in the soaking pit is known as the soaking time of the ingot. The batch processing time in the soaking pit is the longest soaking times of the jobs assigned to the batch. During the time period between steelmaking and soaking, ingots are cast and remain on platforms, which can be moved along rail tracks in the plant. Therefore, this time period is called the track time (Lee, 1979). As the track time increases, the temperature of the ingot decreases, and the required soaking time increases. Depending on the size and shape of the ingot, the increasing rate for the soaking time is different. Patel et al. (1976) provide formulae of the relationship between the soaking time and the track time for three different ingot types. Although the relationship contains a quadratic term, its coefficient is very small, and the relationship is very close to a linear one. Using these relationships and the track times of ingots provided in Lee (1979), ignoring the duplicates and those with track times longer than an eight-hour shift, we calculate the corresponding soaking times and draw the data points in Figure 2. We further fit a line for the points of each ingot type with the restriction that these three lines must meet at a point on the horizontal axis. The resulting lines are also shown in the figure, which can be clearly observed as a good fit to the points. Note that the starting time of each ingot in the soaking process is the time when the track time ends, e.g., the horizontal coordinate of the corresponding point in Figure 2. As a result, the soaking time of a job can be regarded as a proportional-linear increasing function of its starting time. Consequently, we study a two-agent scheduling problem with time-dependent proportional-linear deteriorating job processing times on a bounded parallel-batching machine. Moreover, we extend the problems on a bounded batching machine to an unbounded batching machine. For the unbounded model, there is no upper bound on the number of jobs in the same batch. In the following, we will present an example to illustrate the two-agent scheduling model with time-dependent proportional-linear deteriorating job processing times on a bounded parallel-batching machine. In this example, we assume that the hot rolling mill needs 3 steel slabs, which correspond to 3 ingots and can be regarded as the job set of agent A, i.e., $J^A = \{J_1^A, J_2^A, J_3^A\}$, and customers need 2 steel products, which correspond to 2 ingots and can be regarded as the job set of agent B, i.e., $J^B = \{J_1^B, J_2^B\}$. The actual processing time of job J_j^X is $p_j^X = \alpha_j^X(a + bt)$, $X \in \{A,\ B\}$, where $\alpha_j^X > 0$ is the normal processing time, $a \geq 0$ and $b > 0$ are constants, and t is the starting time of J_j^X. In this example, the job normal processing times are $\alpha_1^A = 1, \alpha_2^A = 2, \alpha_3^A = 3; \alpha_1^B = 0.5, \alpha_2^B = 1$. $a = b = 1$. The capacity of the soaking pit is 2. We further assume that all jobs have the same release dates $t_0 = 1$ and the jobs of different agents cannot be processed in the same batch. The objective is to minimize the makespan $C_{\max}^A$ of agent A, with the restriction that the makespan $C_{\max}^B$ of agent B cannot exceed a given bound $Q_B = 32$. Introduction of the specific parameters can be seen in Section 2. We can obtain an optimal schedule π as shown in Figure 3. The minimum makespan of agent A is 15 and the makespan of agent B is $31 < 32$. This optimal schedule not only satisfies the customers, but also improves the machine efficiency of the next production stage.

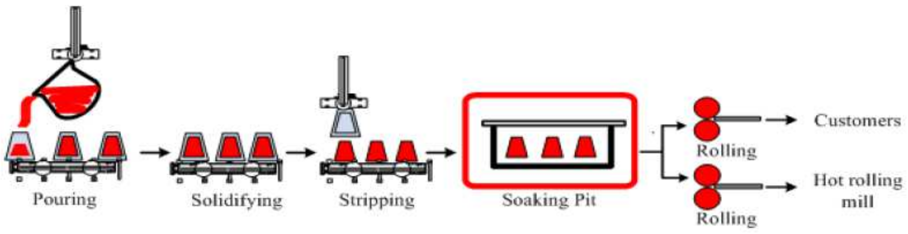

Figure 1. Ingot production processing

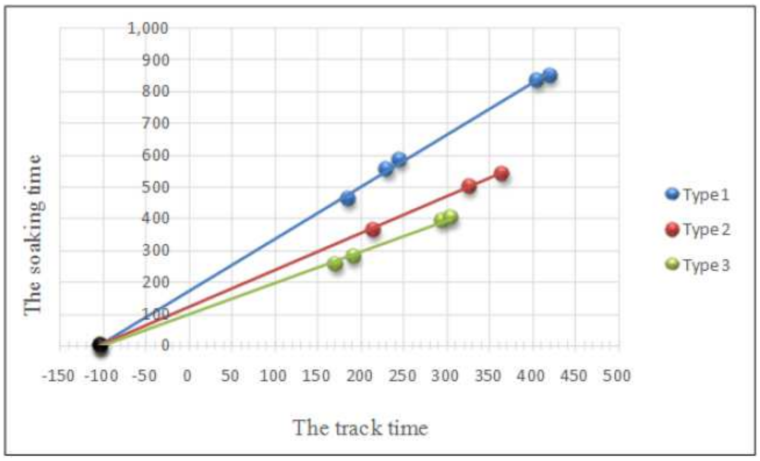

Figure 2. The relationships between the soaking time and the track time

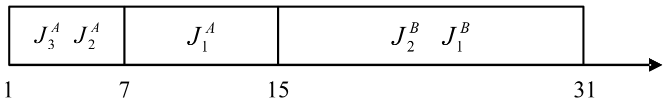

Figure 3. Schedule π

Agent scheduling has become a popular topic in the scheduling literature. Baker and Smith (2003) and Agnetis et al. (2004) first introduce the scheduling problems with two agents, in which all jobs have identical release dates. Baker and Smith (2003) consider a linear combination of the objectives of two agents on a single machine, while Agnetis et al. (2004) study the constrained optimization problems and the Pareto-optimization problems on a single machine or in two-machine shop settings. Leung et al. (2010) generalize the results of Agnetis et al. (2004) and extend the models from a single machine to parallel machines, where the jobs are allowed to preempt and have different release dates. For agent scheduling problems, we refer the reader to the recent survey by Perez-Gonzalez and Framinan (2014) and the recent book by Agnetis et al. (2014). Though agent scheduling has been extensively studied in recent years, results of the research focus mainly on agent scheduling with fixed job processing times, see Chapters 3-5 of the book by Agnetis et al. (2014) for details. At the same time, most agent scheduling problems concern jobs with the same release dates on single machine.

The parallel-batching scheduling problems have attracted wide attention in the field of scheduling research. A parallel-batching machine can process several jobs simultaneously. The processing time of a batch is equal to the longest processing time of the jobs in the batch. All the jobs of the same batch start and complete at the same time. Once processing of a batch begins, it cannot be interrupted, and other jobs cannot be added to the batch. According to the capacity of the batching machine, the problem may be viewed as an unbounded model or a bounded model. Potts and Kovalyov (2000) present a review of scheduling problems with batching.

To the best of our knowledge, there are only two pieces of previous work considering constrained optimization of two-agent scheduling on a parallel-batching machine, in which the jobs have fixed processing times and the same release dates. Li and Yuan (2012) study the scheduling problems with two agents on an unbounded parallel-batching machine. They provide polynomial or pseudo-polynomial time algorithms to solve various combinations of regular objective functions for incompatible and compatible agents. Fan et al. (2013) consider the scheduling problems with two agents on a bounded parallel-batching machine. They focus on minimizing the makespan or the total completion time of one agent, subject to an upper bound on the makespan of the other agent for incompatible and compatible agents.

Scheduling problems with time-dependent job processing times have received considerable attention. For these problems, we refer the reader to the survey by Cheng et al. (2004) and the book by Gawiejnowicz (2008). However, the jobs in these time-dependent scheduling works are individual jobs without agents or may be considered as all belonging to one single agent. In this paper, we focus on the scheduling problems with time-dependent proportional-linear deteriorating job processing times and the jobs belonging to two different agents.

Contrary to agent scheduling with fixed job processing times, research on agent scheduling with time-dependent deteriorating job processing times is very limited. Liu and Tang (2008) consider the two-agent single machine scheduling problems with proportional deterioration. However, they only focus on the criteria of maximum lateness, makespan, total completion time and maximum cost function. Liu et al. (2010) study the two-agent group scheduling problems with proportional deterioration and proportional-linear deterioration on a single machine. The objective is to minimize the total completion time of the first agent, with the restriction that the maximum cost of the second agent cannot exceed a given upper bound. Gawiejnowicz et al. (2011) consider the two-agent single machine scheduling problem with proportional deterioration. The objective is to minimize the total tardiness of the first agent, with the constraint that no tardy job is allowed for the second agent. Gawiejnowicz and Suwalski (2014) first study the two-agent single-machine scheduling problem with proportional deterioration including a non-trivial NP-completeness proof. They propose an exact algorithm and a meta-heuristic for minimizing a weighted sum of the total weighted completion time of one agent and the maximum lateness of the other agent. Liu et al. (2011) consider two-agent single-machine scheduling problems with proportional-linear deterioration to minimize the objective function of one agent while limiting the objective value of the other agent. They study two problems with different combinations of the objective functions and present optimal polynomial time algorithms to solve them. Yin et al. (2015) extend the research of Liu et al. (2011) to other combinations of the two agents' objective functions. He and Leung (2016) also consider the two-agent single-machine scheduling problem with proportional-linear deteriorating and proportional-linear shortening job processing times. However, all of the above mentioned works consider only jobs with the same release dates. Agnetis et al. (2014, Section 6.2) provide a detailed discussion of agent scheduling problems with time-dependent deteriorating job processing times.

Although both the two-agent scheduling problems with time-dependent deteriorating job processing times and the implementation of two-agent scheduling problems on a parallel-batching machine have been extensively studied in the literature, we have not identified any previous research reports on an integrated problem with all three features of two-agent, time-dependent deteriorating job processing times and parallel-batching machine altogether. In this paper we study the two-agent scheduling problems on a single parallel-batching machine, in which the processing time of a job is a proportional-linear increasing function of its starting time. Because the single-agent problems and the integrated problems with any of the above two features are only special cases of our problems, our problems are much more complex and more difficult than the previous work on these problems. Because of the deteriorating jobs and the two-agent scheduling in our problems, many solution methods for the previous two-agent scheduling problems with fixed job processing times and for the single-agent scheduling problems are no longer suitable for our problems. We exploit the structure of the problem and present optimal properties for different versions of the problem. Based on these properties, we design optimal solutions for the solvable problems. In addition, we show that some problems are NP-hard and develop polynomial or pseudo-polynomial time algorithms to solve certain special cases of these intractable problems. Finally, we generalize the integrated problem to jobs with different release dates. There is no previous result for this scenario even for the version with fixed processing times. We analyze the computational complexity for these problems and establish the foundation of theoretical research for the two-agent scheduling problems with different release dates. For the example case of the ingot soaking problem in the steel industry mentioned earlier, the effective solution of this integrated scheduling problem can guide the production and save energy in the soaking process, as well as improve customer satisfaction.

The rest of the paper is organized as follows. In Section 2, we specify our notation and provide an overview of the results to be presented in later sections. Section 3 presents the results of the unbounded batching machine scheduling problems. Section 4 discusses the results of the problems on the bounded batching machine. Finally, Section 5 provides conclusions.

2. Notation and overview of problems studied

The problem studied here can be described as follows. There are two agents A and B. Each of them generates a set of jobs $J^A = \{J_1^A,\ J_2^A,\ \ldots,\ J_{n_A}^A\}$ and $J^B = \{J_1^B,\ J_2^B,\ \ldots,\ J_{n_B}^B\}$, respectively. Let n be the total number of jobs, where $n = n_A + n_B$. The jobs will be processed nonpreemptively on a common batching machine. The batching machine can process up to c jobs simultaneously as a batch. We assume that the actual processing time of job J_j^X is $p_j^X = \alpha_j^X(a + bt)$, $X \in \{A,\ B\}$, where $\alpha_j^X > 0$ is the normal processing time of job J_j^X for $j = 1, 2, \ldots, n_X$, $a \geq 0$ and $b > 0$ are constants, and t is the starting time of J_j^X. Let $d_j^X \geq 0$ be the due date of job J_j^X. We assume that either all jobs are simultaneously available for processing at time $t_0 \geq 0$, or each job J_j^X has an individual release date $r_j^X > 0$. Let $M = (t_0 + \frac{a}{b})\prod_{h=1}^{n_A}(1 + \alpha_h^A)\prod_{k=1}^{n_B}(1 + \alpha_k^B) - \frac{a}{b}$. In a given schedule we denote the completion time of job J_j^X as C_j^X. We use U_j^X to indicate whether job J_j^X is tardy. $U_j^X = 1$, if job J_j^X is tardy, and $U_j^X = 0$, otherwise.

We consider various scheduling problems denoted by the classification scheme $\alpha|\beta|\gamma^A : \gamma^B$ (Agnetis et al., 2004), where α indicates the scheduling environment, β denotes the additional constraints on the jobs, and $\gamma^A : \gamma^B$ defines the objective function γ^A of agent A to be minimized subject to the objective function γ^B of agent B not exceeding a given value $Q_B \geq 0$. In this paper, we consider one machine problems, implying that $\alpha = 1$. Under β, we use $p_j^X = \alpha_j^X(a+bt)$ to indicate the actual processing time of job J_j^X, $j = 1, 2, \ldots, n_X$; $p - batch$ represents the parallel-batching machine; $c = \infty$ and $c < n$ represent the unbounded and bounded capacity of the parallel-batching machine, respectively; IF and CF represent incompatible and compatible agents, respectively, where the incompatible agents mean that one batch contains only jobs from the same agent and the compatible agents mean that one batch may contain jobs from different agents; r_j^X implies that each job has an individual release date and this parameter is omitted if the release dates are equal. We mainly consider the minimization of the following objectives: the number of tardy jobs $\sum U_j$, the makespan $C_{\max} = \max\{C_j\}$, the maximum of regular functions $f_{\max} = \max\{f_j(C_j)\}$, where $f_j(\cdot)$ is a nondecreasing function of the completion time of job J_j and its value can be calculated in constant time for any one job completion time, and the sum of regular functions $\sum f_j = \sum f_j(C_j)$.

3. The unbounded parallel-batching model

In this section, we consider the unbounded parallel-batching scheduling problems. The jobs may have identical or different release dates. We propose algorithms for the problems and show their complexities based on the compatibility of the two agents. Without loss of generality, we assume that the job parameters are non-negative integers.

3.1. The unbounded parallel-batching with identical release dates

We first present the following two properties for the optimal schedules corresponding to different agent compatibility types respectively. The first result characterizes the sequence of jobs of each agent in an optimal schedule for the incompatible agents. The second result characterizes the sequence of jobs of two agents in an optimal schedule for the compatible agents. These properties can be proved by showing that any schedule violating the property can be improved, or at least does not become worse, by shifting jobs to make it satisfy the property. We omit the details of the proof.

Lemma 3.1. There is an optimal schedule for the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty; IF|\gamma^A : \gamma^B$, where the jobs (batches) from each agent are processed in the shortest normal processing time (SNPT) order.

We refer to a schedule that satisfies the property in Lemma 3.1 as a *D-SNPT-batch* schedule.

Based on Lemma 3.1, if there are two jobs J_i^B and J_j^B with $\alpha_i^B \le \alpha_j^B$ and $d_i^B \ge d_j^B$, then the job J_i^B can be moved to the same batch as the job J_j^B. Hence, for the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty; IF|\gamma^A : f_{\max}^B$, we may assume that the jobs of agent B have been re-indexed such that $\alpha_1^B < \ldots < \alpha_{n_B}^B$ and $\bar{d}_1^B < \ldots < \bar{d}_{n_B}^B$, where $\bar{d}_k^B$ is an induced deadline and can be calculated in constant time, such that $f_k^B(C_k^B) \le Q_B$ for $C_k^B \le \bar{d}_k^B$ and $f_k^B(C_k^B) > Q_B$ for $C_k^B > \bar{d}_k^B$.

Lemma 3.2. There is an optimal schedule for the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty; CF|\gamma^A : \gamma^B$, where the jobs (batches) from the two agents are processed in the SNPT order.

We refer to a schedule that satisfies the property in Lemma 3.2 as an *SNPT-batch* schedule.

3.1.1. $1|p-batch; p_j^X = \alpha_j^X(a+bt); c=\infty|\sum f_j^A : f_{\max}^B$

Li and Yuan (2012) propose dynamic programming algorithms for $1|p-batch; c=\infty|\sum f_j^A : f_{\max}^B$ with fixed processing times for the incompatible and compatible cases. To solve the problems with proportional-linear deteriorating processing times, the running times of these algorithms would be exponential. Hence, in this subsection, we present new dynamic programming algorithms for the problem based on the compatibility of two agents. Let $\Delta = \max\{f_j(M) : 1 \le j \le n\}$. For the incompatible case, we propose the following algorithm with overall time complexity of $O(n_A^3 n_B n^2 \Delta^2)$.

Algorithm DP1

Based on Lemma 3.1, we may assume that the jobs of each agent are indexed according to the SNPT rule, i.e., $\alpha_1^A \le \alpha_2^A \ldots \le \alpha_{n_A}^A$ and $\alpha_1^B \le \alpha_2^B \ldots \le \alpha_{n_B}^B$. Define $C(h_A, k_B, t_A)$ as the minimum completion time of a partial *D-SNPT-batch* schedule containing jobs $J_1^A, J_2^A, \ldots, J_{h_A}^A$, $J_1^B, J_2^B, \ldots, J_{k_B}^B$ such that the objective value of agent A is exactly t_A and no job of agent B completes after its induced deadline. If there is no feasible schedule, we define $C(h_A, k_B, t_A) = +\infty$. The initial condition is $C(0,0,t_A) = \begin{cases} t_0, & \text{if } t_A = 0, \\ +\infty, & \text{otherwise.} \end{cases}$

In a feasible schedule, assume that the last batch comes from agent A and is of the form $J_{l_A}^A, \ldots, J_{h_A}^A$ with $1 \le l_A \le h_A$. Then we have $C(h_A, k_B, t_A) = (1+b\alpha_{h_A}^A)C(l_A-1, k_B, t_A^*) + a\alpha_{h_A}^A$, where $t_A^* = t_A - \sum_{i=l_A}^{h_A} f_i^A(C(h_A, k_B, t_A))$. Assume that the last batch comes from agent B and is of the form $J_{l_B}^B, \ldots, J_{k_B}^B$ with $1 \le l_B \le k_B$. If $(1+b\alpha_{k_B}^B)C(h_A, l_B-1, t_A) + a\alpha_{k_B}^B \le \bar{d}_{l_B}^B$, where $\bar{d}_{l_B}^B$ is the induced deadline such that $f_{l_B}^B((1+b\alpha_{k_B}^B)C(h_A, l_B-1, t_A) + a\alpha_{k_B}^B) \le Q_B$, then we have $C(h_A, k_B, t_A) = (1+b\alpha_{k_B}^B)C(h_A, l_B-1, t_A) + a\alpha_{k_B}^B$.

Summarizing the above analysis, we have the following recursive relation:

$$C(h_A, k_B, t_A) = \min \begin{cases} (1+b\alpha_{h_A}^A) \min\limits_{1 \le l_A \le h_A} \min\limits_{t_A^* \in \Gamma} C(l_A-1, k_B, t_A^*) + a\alpha_{h_A}^A, \\ (1+b\alpha_{k_B}^B) \min\limits_{1 \le l_B \le k_B} C(h_A, l_B-1, t_A) + a\alpha_{k_B}^B, \\ \qquad\qquad\qquad\qquad \text{if } (1+b\alpha_{k_B}^B)C(h_A, l_B-1, t_A) + a\alpha_{k_B}^B \le \bar{d}_{l_B}^B. \end{cases}$$

where $\Gamma = \{t_A^* : t_A^* + \sum_{i=l_A}^{h_A} f_i^A((1+b\alpha_{h_A}^A)C(l_A-1, k_B, t_A^*) + a\alpha_{h_A}^A) = t_A\}$.

The value of t_A belongs to $[0, n\Delta]$. There are $n_A n_B n\Delta$ states in the dynamic program. In each recursion, the value of t_A^* has at most $n\Delta$ choices, and for each t_A^*, we need $O(n_A)$ time to check whether $t_A^* \in \Gamma$ or not. Hence, all $C(h_A, k_B, t_A)$ can be calculated in $O(n_A^3 n_B n^2 \Delta^2)$ time.

The optimal solution value is $\min\{t_A \in [0, n\Delta] : C(n_A, n_B, t_A) < +\infty\}$.

Theorem 3.1.1. The problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c=\infty; IF|\sum f_j^A : f_{\max}^B$ can be solved in $O(n_A^3 n_B n^2 \Delta^2)$ time.

For the compatible case, we propose the following algorithm for the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c=\infty; CF|\sum f_j^A : f_{\max}^B$.

Algorithm DP2

Based on Lemma 3.2, we may assume that the jobs of two agents are indexed according to the SNPT rule, i.e., $\alpha_1 \le \alpha_2 \le \cdots \le \alpha_n$. Define $C(j, t_A)$ as the minimum completion time of a partial *SNPT-batch* schedule containing jobs $J_1, J_2, \ldots, J_j$ such that the objective value of agent A is exactly t_A and no job of agent B completes after its induced deadline. If there is no feasible schedule, we define $C(j, t_A) = +\infty$. The initial condition is $C(0, t_A) = \begin{cases} t_0, & \text{if } t_A = 0, \\ +\infty, & \text{otherwise.} \end{cases}$

In a feasible schedule, assume that the last batch is of the form $J_i, \ldots, J_j$ with $1 \le i \le j$. Then we have recursive relation:

$$C(j, t_A) = (1 + b\alpha_j) \min_{1 \le i \le j} \min_{t_A^* \in \Gamma} C(i-1, t_A^*) + a\alpha_j,$$

where $\Gamma = \{t_A^* : t_A^* + \sum\limits_{l=i}^{j} f_l((1+b\alpha_j)C(i-1, t_A^*) + a\alpha_j) = t_A\}$,

$$f_l((1+b\alpha_j)C(i-1,t_A^*)+a\alpha_j) = \begin{cases} f_l((1+b\alpha_j)C(i-1,t_A^*)+a\alpha_j), & \text{if } J_l \in J^A, \\ 0, & \text{if } J_l \in J^B \text{ and } (1+b\alpha_j)C(i-1,t_A^*)+a\alpha_j \le \bar{d}_l^B, \\ +\infty, & \text{if } J_l \in J^B \text{ and } (1+b\alpha_j)C(i-1,t_A^*)+a\alpha_j > \bar{d}_l^B. \end{cases}$$

where $\bar{d}_l^B$ is the induced deadline such that $f_l((1+b\alpha_j)C(i-1,t_A^*)+a\alpha_j) \le Q_B$ if $J_l \in J^B$. The optimal solution value is $\min\{t_A \in [0, n\Delta] : C(n, t_A) < +\infty\}$.

Theorem 3.1.2. The problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c=\infty; CF|\sum f_j^A : f_{\max}^B$ can be solved in $O(n^5\Delta^2)$ time.

3.1.2. $1|p-batch; p_j^X = \alpha_j^X(a+bt); c=\infty|\sum f_j^A : \sum f_j^B$

In this subsection, we consider another problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c=\infty|\sum f_j^A : \sum f_j^B$ based on the compatibility. We also propose new dynamic programming algorithms for the incompatible and compatible cases. Their computational complexities are $O(n_A n_B n \Delta Q_B(n_A^2 n\Delta + n_B^2 Q_B))$ and $O(n^5\Delta^2 Q_B^2)$, respectively.

Algorithm DP3

Based on Lemma 3.1, we may assume that the jobs of each agent are indexed according to the SNPT rule, i.e., $\alpha_1^A \leq \alpha_2^A \ldots \leq \alpha_{n_A}^A$ and $\alpha_1^B \leq \alpha_2^B \ldots \leq \alpha_{n_B}^B$. Define $C(h_A, k_B, t_A, t_B)$ as the minimum completion time of a partial *D-SNPT-batch* schedule containing jobs $J_1^A, J_2^A, \ldots, J_{h_A}^A$, $J_1^B, J_2^B, \ldots, J_{k_B}^B$ such that the objective values of agent A and agent B are exactly t_A and t_B, respectively, and $t_B \leq Q_B$. If there is no feasible schedule, we define $C(h_A, k_B, t_A, t_B) = +\infty$. The initial condition is $C(0, 0, t_A, t_B) = \begin{cases} t_0, & \text{if } t_A = 0 \text{ and } t_B = 0, \\ +\infty, & \text{otherwise.} \end{cases}$

Similar to DP1, we have the following recursive relation:

$$C(h_A, k_B, t_A, t_B) = \min \begin{cases} (1 + b\alpha_{h_A}^A) \min\limits_{1 \leq l_A \leq h_A} \min\limits_{t_A^* \in \Gamma} C(l_A - 1, k_B, t_A^*, t_B) + a\alpha_{h_A}^A, \\ (1 + b\alpha_{k_B}^B) \min\limits_{1 \leq l_B \leq k_B} \min\limits_{t_B^* \in \Gamma'} C(h_A, l_B - 1, t_A, t_B^*) + a\alpha_{k_B}^B. \end{cases}$$

where $\Gamma = \{t_A^* : t_A^* + \sum\limits_{i=l_A}^{h_A} f_i^A((1 + b\alpha_{h_A}^A)C(l_A - 1, k_B, t_A^*, t_B) + a\alpha_{h_A}^A) = t_A\}$,

$\Gamma' = \{t_B^* : t_B^* + \sum\limits_{i=l_B}^{k_B} f_i^B((1 + b\alpha_{k_B}^B)C(h_A, l_B - 1, t_A, t_B^*) + a\alpha_{k_B}^B) = t_B\}$.

The values of t_A and t_B belong to $[0, n\Delta]$ and $[0, Q_B]$, respectively. There are $n_A n_B n\Delta Q_B$ states in the dynamic program. In each recursion, the value of t_A^* has at most $n\Delta$ choices and t_B^* has at most Q_B choices, and we need $O(n_A)$ and $O(n_B)$ times to check whether $t_A^* \in \Gamma$ and $t_B^* \in \Gamma'$ or not, respectively. Hence, all $C(h_A, k_B, t_A, t_B)$ can be calculated in $O(n_A n_B n\Delta Q_B(n_A^2 n\Delta + n_B^2 Q_B))$ time. The optimal solution value is $\min\{t_A \in [0, n\Delta] : C(n_A, n_B, t_A, t_B) < +\infty, t_B \leq Q_B\}$.

Theorem 3.1.3. The problem $1|p-batch; p_j^X = \alpha_j^X(a + bt); c = \infty; IF|\sum f_j^A : \sum f_j^B$ can be solved in $O(n_A n_B n\Delta Q_B(n_A^2 n\Delta + n_B^2 Q_B))$ time.

Algorithm DP4

Based on Lemma 3.2, we may assume that the jobs of two agents are indexed according to the SNPT rule, i.e., $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_n$. Define $C(j, t_A, t_B)$ as the minimum completion time of a partial *SNPT-batch* schedule containing jobs $J_1, J_2, \ldots, J_j$ such that the objective values of agent A and agent B are exactly t_A and t_B, respectively, and $t_B \leq Q_B$. If there is no feasible schedule, we define $C(j, t_A, t_B) = +\infty$.

The initial condition is $C(0, t_A, t_B) = \begin{cases} t_0, & \text{if } t_A = 0 \text{ and } t_B = 0, \\ +\infty, & \text{otherwise.} \end{cases}$

In a feasible schedule, assume that the last batch is of the form $J_i, \ldots, J_j$ with $1 \le i \le j$. Then we have recursive relation:

$$C(j, t_A, t_B) = (1 + b\alpha_j) \min_{1 \le i \le j} \min_{t_A^* \in \Gamma, t_B^* \in \Gamma'} C(i-1, t_A^*, t_B^*) + a\alpha_j,$$

where $\Gamma = \{t_A^* : t_A^* + \sum_{l=i}^{j} f_l^A((1+b\alpha_j)C(i-1, t_A^*, t_B^*) + a\alpha_j) = t_A, J_l \in J^A\}$,

$\Gamma' = \{t_B^* : t_B^* + \sum_{l=i}^{j} f_l^B((1+b\alpha_j)C(i-1, t_A^*, t_B^*) + a\alpha_j) = t_B, J_l \in J^B\}$.

The optimal solution value is $\min\{t_A \in [0, n\Delta] : C(n, t_A, t_B) < +\infty, t_B \le Q_B\}$.

Theorem 3.1.4. The problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty; CF|\sum f_j^A : \sum f_j^B$ can be solved in $O(n^5\Delta^2 Q_B^2)$ time.

3.1.3. $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty|f_{\max}^A : f_{\max}^B$, $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty|\sum U_j^A : \sum U_j^B$ *and* $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty|\sum U_j^A : f_{\max}^B$

In this subsection, we generalize the algorithms proposed by Li and Yuan (2012) for their previous problem with fixed processing times to our problems with time-dependent deteriorating job processing times, we can obtain the corresponding extended results for the new problems with time-dependent deteriorating job processing times. Because of space limit, we omit the details here. If the readers want to read more information, please refer to the paper by Li and Yuan (2012).

Remark 1. For each fixed value Q_A, whether a feasible schedule exists for the incompatible and compatible cases of decision problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty|L_{\max}^A \le Q_A : f_{\max}^B$ can be determined in $O(nn_An_B)$ time and $O(n^3)$ time, respectively.

Remark 2. The incompatible and compatible cases of problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty|L_{\max}^A : f_{\max}^B$ can be solved in $O(nn_An_B \log M)$ time and $O(n^3 \log M)$ time, respectively.

Remark 3. The incompatible and compatible cases of problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty|f_{\max}^A : f_{\max}^B$ can be solved in $O(nn_An_B \log(Q_U - Q_L))$ time and $O(n^3 \log(Q_U - Q_L))$ time, respectively, where $Q_L = \min\{f_j^A(t_0) : 1 \le j \le n_A\}$ and $Q_U = \max\{f_j^A(M) : 1 \le j \le n_A\}$.

Remark 4. The incompatible and compatible cases of problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty|\sum U_j^A : \sum U_j^B$ can be solved in $O(n_A^2 n_B^2 n^2)$ time and $O(n^2 n_A n_B)$ time, respectively.

Remark 5. The incompatible and compatible cases of problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c = \infty|\sum U_j^A : f_{\max}^B$ can be solved in $O(n_A^2 n_B n^2)$ time and $O(n^2 n_A)$ time, respectively.

3.2. The unbounded parallel-batching with distinct release dates

3.2.1. $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c=\infty|C_{\max}^A : C_{\max}^B$

In this subsection, we show that the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c=\infty|C_{\max}^A : C_{\max}^B$ can be solved in polynomial time for both incompatible and compatible cases.

3.2.1.1. $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c=\infty; IF|C_{\max}^A : C_{\max}^B$.

In this subsection, we propose a polynomial time algorithm for the incompatible case. We first design a polynomial time dynamic programming algorithm to determine whether a feasible schedule exists for the decision problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c=\infty; IF|C_{\max}^A \le Q_A : C_{\max}^B$. The dynamic programming is based on the following properties of an optimal schedule for the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c=\infty; IF|C_{\max}^A : C_{\max}^B$.

Lemma 3.2.1. For the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c=\infty; IF|C_{\max}^A : C_{\max}^B$, there is an optimal batch sequence $\pi = (B_1, B_2, \ldots, B_m)$ such that if two jobs J_i and J_j belong to the same agent and distinct batches, with $J_i \in B_x$, $J_j \in B_y$ and $x < y$, then $\alpha_i > \alpha_j$.

Corollary 3.2.2. There is an optimal batch sequence $\pi = (B_1, B_2, \ldots, B_m)$ for the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c=\infty; IF|C_{\max}^A : C_{\max}^B$ such that each batch B_x of agent A or B is in the form $B_x = \{J_j \in J^X : \alpha_l \le \alpha_j \le \alpha_u\}$ for some numbers l and u.

If two jobs J_i and J_j are from the same agent with $r_i \le r_j$ and $\alpha_i \le \alpha_j$, then we can always put J_i in the same batch as J_j without increasing the makespan of each agent. Thus, we can delete J_i from the job set. Therefore, without loss of generality, we can re-index the jobs of each agent such that $r_1^X < r_2^X < \cdots < r_{n_X}^X$ and $\alpha_1^X > \alpha_2^X > \cdots > \alpha_{n_X}^X$ for $X = A, B$.

Based on the above properties, we present the following dynamic programming algorithm to determine whether a feasible schedule exists for the decision problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c=\infty; IF|C_{\max}^A \le Q_A : C_{\max}^B$. Then we use this algorithm as a subroutine to solve the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c=\infty; IF|C_{\max}^A : C_{\max}^B$.

Algorithm DP5

Given any fixed $Q_A > 0$, define $f_{Q_A}(h_A, k_B)$ as the minimum makespan of a partial schedule containing jobs $J_1^A, J_2^A, \ldots, J_{h_A}^A, J_1^B, J_2^B, \ldots, J_{k_B}^B$ such that the completion time of the last job of agent A is at most Q_A and the completion time of the last job of agent B is at most Q_B. If there is no feasible schedule, we define $f_{Q_A}(h_A, k_B) = +\infty$. The initial condition is $f_{Q_A}(0,0) = 0$.

In a feasible schedule, assume that the last batch belongs to agent A and it is of the form $\{J_{l_A+1}^A, J_{l_A+2}^A, \ldots, J_{h_A}^A\}$ with $l_A < h_A$. Then we have

$f_A = \min_{0 \le l_A \le h_A - 1} \{\max\{f_{Q_A}(l_A, k_B), r^A_{h_A}\} \cdot (1 + b\alpha^A_{l_A+1}) + a\alpha^A_{l_A+1},$ if $\max\{f_{Q_A}(l_A, k_B), r^A_{h_A}\} \cdot (1 + b\alpha^A_{l_A+1}) + a\alpha^A_{l_A+1} \le Q_A\}$.

Assume that the last batch belongs to agent B and it is of the form $\{J^B_{l_B+1}, J^B_{l_B+2}, \dots, J^B_{k_B}\}$ with $l_B < k_B$. Then we have

$f_B = \min_{0 \le l_B \le k_B - 1} \{\max\{f_{Q_A}(h_A, l_B), r^B_{k_B}\} \cdot (1 + b\alpha^B_{l_B+1}) + a\alpha^B_{l_B+1},$ if $\max\{f_{Q_A}(h_A, l_B), r^B_{k_B}\} \cdot (1 + b\alpha^B_{l_B+1}) + a\alpha^B_{l_B+1} \le Q_B\}$.

Hence, the recursive relation is $f_{Q_A}(h_A, k_B) = \min\{f_A, f_B\}$.

In the above recursive relation, if $f_{Q_A}(n_A, n_B) < +\infty$, then we can obtain a feasible schedule for the decision problem $1|p - batch; p^X_j = \alpha^X_j(a + bt); r^X_j; c = \infty; IF|C^A_{\max} \le Q_A : C^B_{\max}$ in $O(n_A n_B n)$ time.

Theorem 3.2.3. The problem $1|p - batch; p^X_j = \alpha^X_j(a + bt); r^X_j; c = \infty; IF|C^A_{\max} : C^B_{\max}$ can be solved in $O(n_A n_B n \log(Q'_U - Q'_L))$ time, where Q'_U and Q'_L are upper and lower bounds of Q_A, respectively.

Proof. For each $Q_A > 0$, we can use DP5 as a subroutine to find a feasible schedule such that $C^A_{\max} \le Q_A$ and $C^B_{\max} \le Q_B$. Observe that a lower bound for Q_A is $Q'_L = F(n_A)$ which is the optimal value for the single-agent problem $1|p - batch; p_j = \alpha_j(a + bt); r_j; c = \infty|C_{\max}$ that only schedules the jobs $J^A_1, J^A_2, \dots, J^A_{n_A}$ and that can be solved using a dynamic programming algorithm similar to DP^1 of Li et al. (2011) in $O(n^2_A)$ time. An upper bound for Q_A is $Q'_U = F'(n_A)$ which is the optimal value for the problem $1|p - batch; p^A_j = \alpha^A_j(a + bt); r^A_j; c = \infty; FB|C^A_{\max}$, where FB means forbidden interval. Here "FB" refers to the processing intervals for only optimally scheduling the jobs $J^B_1, J^B_2, \dots, J^B_{n_B}$ for the problem $1|p - batch; p^B_j = \alpha^B_j(a + bt); r^B_j; c = \infty|C^B_{\max}$. Then $F'(n_A)$ can be optimally solved in $O(n^2_B + n^2_A n_B)$ time on the basis of the problem $1|p - batch; r_j; c = \infty; FB|C_{\max}$ (Yuan et al., 2008). We can conduct a binary search in the range $[Q'_L, Q'_U]$ to determine the optimal value $Q^*_A = \min\{Q_A : f_{Q_A}(n_A, n_B) < +\infty\}$. Hence, the problem $1|p - batch; p^X_j = \alpha^X_j(a + bt); r^X_j; c = \infty; IF|C^A_{\max} : C^B_{\max}$ can be solved in $O(n_A n_B n \log(Q'_U - Q'_L))$. $\square$

3.2.1.2. $1|p - batch; p^X_j = \alpha^X_j(a + bt); r^X_j; c = \infty; CF|C^A_{\max} : C^B_{\max}$.

In view of Lemma 3.2.1, we may assume that the jobs have been indexed such that $r_1 < \cdots < r_n$ and $\alpha_1 > \cdots > \alpha_n$. We define a batch as *X-pure* if this batch contains only the jobs of agent X, $X = A, B$. We have the following properties for the compatible case.

Lemma 3.2.4. For the problem $1|p - batch; p^X_j = \alpha^X_j(a + bt); r^X_j; c = \infty; CF|C^A_{\max} : C^B_{\max}$, there is an optimal schedule in the form (π_1, π_2, π_3) that has the following properties:

1) the partial schedule π_2 contains only all the *B-pure* batches, π_3 contains part of *A-pure* batches (if any), and π_1 contains the remaining batches;
2) all the jobs (batches) in the partial schedules π_1 and π_2 are scheduled in decreasing order of their normal processing times, all the jobs (batches) in the partial schedules π_1 and π_3 are also scheduled in decreasing order of their normal processing times.

Lemma 3.2.4 implies that each batch contains only consecutive jobs.

Algorithm 1

Based on Lemmas 3.2.1 and 3.2.4, we may assume that the jobs have been indexed as follows: $J_1^X, \ldots, J_{i-1}^X, J_i^A, J_{i+1}^B, \ldots, J_k^B, J_{k+1}^A, \ldots, J_n^A$, $X \in \{A, B\}$, such that $r_1 < \cdots < r_n$ and $\alpha_1 > \cdots > \alpha_n$, where the job J_i^A is the last job of agent A in π_1, the jobs $J_{i+1}^B, \ldots, J_k^B$ belong to agent B, the job J_k^B is the last job of agent B, the jobs $J_{k+1}^A, \ldots, J_n^A$ belong to agent A. Let $F(j)$ be the minimum completion time of a partial schedule of jobs $J_1^X, \ldots, J_j^X$, $X \in \{A, B\}$. Using the solution method for the single-agent problem $1|p-batch; p_j = \alpha_j(a+bt); r_j; c = \infty|C_{\max}$, which takes $O(n^2)$ time, we can obtain the minimum completion time $F(j)$ of any one job J_j $j = 1, 2, \ldots, n$ for a given schedule sequence. The minimum completion time of job J_k^B in the partial schedule of jobs $J_1^X, \ldots, J_{i-1}^X, J_i^A, J_{i+1}^B, \ldots, J_k^B$ is denoted by $F_{opt}(k)$.
1) If $F_{opt}(k) \le Q_B$, then we can compute $F(k+1), \ldots, F(n)$ for the job sequence $J_1^X, \ldots, J_{i-1}^X, J_i^A$, $J_{i+1}^B, \ldots, J_k^B, J_{k+1}^A, \ldots, J_n^A$, $X \in \{A, B\}$. This takes $O(n^2)$ time. There are two outcomes:
(1) We find the first job $J_j \in J^A$, for $k+1 \le j \le n$, such that $F(j) > Q_B$. We denote this job by J_{j^*}, i.e., $F(j^*) > Q_B$.

For $j = j^*, j^*+1, \ldots, n$, if the last batch in the partial schedule of jobs $J_1^X, \ldots, J_{i-1}^X, J_i^A$, $J_{i+1}^B, \ldots, J_k^B, \ldots, J_j^A$ contains at least one job of agent B, then we compute $F(j)$ as $F(j) = \min_{j^*-1 \le i \le j-1} \{\max\{F(i), r_j\} \cdot (1 + b\alpha_{i+1}) + a\alpha_{i+1}\}$. The time taken to compute $F(j)$ is $O(n_A)$. Hence, the running time for this case is $O(n_A^2)$.

If the last batch in the partial schedule of jobs $J_1^X, \ldots, J_{i-1}^X, J_i^A, J_{i+1}^B, \ldots, J_k^B, \ldots, J_j^A$ does not contain any one job of agent B, then we compute $F(j)$ as $F(j) = \min_{k \le i \le j-1} \{\max\{F(i), r_j\} \cdot (1 + \alpha_{i+1}) + a\alpha_{i+1}\}$. The time taken to compute $F(j)$ is $O(n_A)$. Hence, the running time for this case is also $O(n_A^2)$.
(2) If $F(j) \le Q_B$ for all $k+1 \le j \le n$, then we move all jobs $J_{k+1}, \ldots, J_n$ to the front of J_k^B, compute $F(k)$ and check whether $F(k) \le Q_B$ or not. If yes, then we move all jobs $J_{k+1}, \ldots, J_n$ to the front of J_{k-1}^B and continue to check whether $F(k) \le Q_B$ or not. If yes, we keep doing this until the last time the minimum completion time of job J_k^B is not greater than Q_B. In this

case, the optimal solution value of agent A is $F(n)$ that can be obtained in $O(n^2 n_B)$ time.

If when all jobs $J_{k+1}, \ldots, J_n$ are moved to the front of J_{i+1}^B and the minimum completion time of job J_k^B is still not greater than Q_B, then the jobs of agent B before the job J_i^A from back to front are moved one-by-one to the position behind the job J_n and are processed in the same sequence of their original positions, i.e., if $J_{i-2}^X, J_{i-1}^X \in J^B$, then we have the schedule $J_1^X, \ldots, J_{i-3}^X, J_i^A, J_{k+1}^A, \ldots, J_n^A, J_{i-2}^X, J_{i-1}^X, J_{i+1}^B, \ldots, J_k^B$. And then compute $F(k)$ and check whether $F(k) \le Q_B$ or not. If yes, we keep doing this until the last time the minimum completion time of job J_k^B is not greater than Q_B. There are at most n_B movements for the jobs of agent B before the job J_i^A. So the running time for this case is also $O(n^2 n_B)$.

Summarizing the above analysis, the optimal solution value of agent A is $F(n)$. The running time of the algorithm for this case is $O(n^2 n_B)$.

2) If $F_{opt}(k) > Q_B$, then the jobs of agent A before the job J_{i+1}^B from back to front are moved one-by-one to the position behind the job J_k^B and are processed in the same sequence of their original positions, i.e., if $J_{i-1}^X \in J^A$, then we have the schedule $J_1^X, \ldots, J_{i-2}^X, J_{i+1}^B, \ldots, J_k^B, J_{i-1}^X, J_i^A, J_{k+1}^A, \ldots, J_n^A$. And then compute $F(k)$ and check whether $F(k) \le Q_B$ or not. If no, we keep doing this until the minimum completion time of job J_k^B is not greater than Q_B. There are at most n_A movements for the jobs of agent A before the job J_{i+1}^B. Hence, in this case we can obtain the optimal solution value $F(n)$ of agent A in $O(n^2 n_A)$ time.

3.2.2. $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c=\infty|\sum U_j^A : \sum U_j^B$

Miao et al. (2012) prove that the problem $1|p-batch; p_j = \alpha_j t; r_j; c=\infty|L_{\max}$ is NP-hard. Based on this result, it is easy to see that the problem $1|p-batch; p_j = \alpha_j t; r_j; c=\infty|\sum U_j$ is also NP-hard. Consider a special case of the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c = \infty|\sum U_j^A : \sum U_j^B$ where $a=0$ and the upper bound Q_B of the objective value $\sum U_j^B$ for agent B is sufficiently large such that the jobs of agent B can be processed at the end of the schedule and start after the completion of all the jobs of agent A, while the upper bound restriction is still satisfied. The problem is then equivalent to the single-agent problem (for agent A) $1|p-batch; p_j = \alpha_j t; r_j; c=\infty|\sum U_j$. Hence, we have the following result.

Theorem 3.2.5. The problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c = \infty|\sum U_j^A : \sum U_j^B$ is NP-hard for both incompatible and compatible cases.

4. The bounded parallel-batching model

In this section, we consider the complexity of the bounded parallel-batching model based on the compatibility of two agents. We can prove that most of the two-agent scheduling problems with deteriorating jobs on a bounded parallel-batching machine are NP-hard. We present optimal solution methods for some solvable special cases.

4.1. The bounded parallel-batching with identical release dates

4.1.1. $1|p-batch; p_j^X = \alpha_j^X(a+bt); c<n|C_{\max}^A : C_{\max}^B$

We first show that $1|p-batch; p_j^X = \alpha_j^X(a+bt); c<n; IF|C_{\max}^A : C_{\max}^B$ is solvable in polynomial time.

Theorem 4.1.1. The problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c<n; IF|C_{\max}^A : C_{\max}^B$ can be solved in $O(n\log n)$ time.

Proof. For each agent, we can obtain an optimal solution in $O(n\log n)$ time by the Full Batch Longest Normal Processing Time (FBLNPT) rule, which is similar to the Algorithm FBLDR proposed by Li et al. (2011). Hence, we can decompose the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c<n; IF|C_{\max}^A : C_{\max}^B$ into two independent subproblems for each agent. Then we can obtain an optimal schedule as follows: all the batches of agent A obtained by the FBLNPT rule are processed first, followed by all the batches of agent B obtained by the FBLNPT rule if $C_{\max}^B \le Q_B$; otherwise all the batches of agent B are processed first, followed by all the batches of agent A. If $C_{\max}^B > Q_B$ in the second schedule, then the problem has no solution. $\Box$

We now show that $1|p-batch; p_j^X = \alpha_j^X(a+bt); c<n; CF|C_{\max}^A : C_{\max}^B$ is NP-hard by a reduction from the Product Partition Problem, which is known to be NP-complete in the strong sense (Ng et al., 2010).

Product Partition (PP) Problem : Given positive integer numbers $a_1, a_2, ..., a_m$, is there a subset $S' \subset S := \{1,2,...,m\}$ such that $\prod_{i\in S'} a_i = \prod_{i\in S\setminus S'} a_i$?

Theorem 4.1.2. The problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c<n; CF|C_{\max}^A : C_{\max}^B$ is NP-hard even if $c=2$.

Proof. The decision version of the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c<n; CF|C_{\max}^A : C_{\max}^B$ is clearly in NP. Given an instance of the PP problem, Let $D = \prod_{i\in S} a_i$, and $H = \sqrt{D}$, we construct an instance of the decision version of our problem as follows:

There are $n_A = 3m$ jobs of m types of agent A and $n_B = m$ jobs of agent B.

Let $a = 0$, $b = 1$.

The normal processing times of A-agent's jobs and B-agent's jobs are defined by

$$\alpha_{i1}^A = H^{4i}a_i - 1, \quad \alpha_{i2}^A = \alpha_{i3}^A = \frac{H^{4i}}{a_i} - 1; \quad \alpha_i^B = H^{4i} - 1; \quad for\ i = 1, 2, \ldots, m.$$

All jobs are simultaneously available at time $t_0 = 1$.
The upper bound Q_B is defined by $Q_B = H^{2(m^2+m)+1}$.
The threshold value of agent A is defined by $Q_A = H^{4(m^2+m)+1}$.

It can be seen that the above reduction from the strong NP-complete PP problem is polynomial with respect to the input problem length. But the magnitude of the resulting problem parameters is not bounded by a polynomial in the length and the magnitude of the PP problem, and so the reduction is not pseudo-polynomial. Consequently we are proving that $1|p - batch; p_j^X = \alpha_j^X(a + bt); c < n; CF|C_{\max}^A : C_{\max}^B$ is NP-hard in ordinary sense. We now show that there is a schedule to this instance of our problem with $C_{\max}^A \le Q_A$ and $C_{\max}^B \le Q_B$ if and only if there is a solution to the PP problem.

If Part. Given a subset $S' \subseteq S$ such that $\prod_{i\in S'} a_i = \prod_{i\in S\backslash S'} a_i$, we construct a schedule for the instance as follows: $B_{i1} = \{J_{i1}^A, J_i^B\}$ for $i \in S'$, $B_{i2} = \{J_{i2}^A, J_i^B\}$ for $i \in S\backslash S'$, $B_{i3} = \{J_{i2}^A, J_{i3}^A\}$ for $i \in S'$, $B_{i4} = \{J_{i1}^A, J_{i3}^A\}$ for $i \in S\backslash S'$. The batches are processed according to the following order: the batches B_{i1} are first scheduled for all $i \in S'$, followed by the batches B_{i2} for all $i \in S\backslash S'$, followed by the batches B_{i3} for all $i \in S'$, and followed by the batches B_{i4} for all $i \in S\backslash S'$. It is easy to show that $C_{\max}^A \le Q_A$ and $C_{\max}^B \le Q_B$.

Only If Part. Given a schedule for the instance with $C_{\max}^A \le Q_A$ and $C_{\max}^B \le Q_B$, we can conclude that each batch contains only two jobs of the same type; i.e., for each i ($1 \le i \le m$), the jobs $J_{i1}^A, J_{i2}^A, J_{i3}^A$ and J_i^B are divided into two batches (Similar to Li et al., 2011, we can obtain this conclusion, so we omit the details here). This implies that the number of batches is exactly $2m$ and the jobs of agent B are assigned to m distinct batches. Since jobs J_{i2}^A and J_{i3}^A are identical for $i = 1, \ldots, m$, there are only two ways to partition the four jobs of type i into two batches, i.e., $\{J_{i1}^A, J_i^B\}$ and $\{J_{i2}^A, J_{i3}^A\}$, or $\{J_{i2}^A, J_i^B\}$ and $\{J_{i1}^A, J_{i3}^A\}$. Assume that the jobs J_{i1}^A and J_i^B are processed in the same batch B_{i1} for $i \in S'$, and the jobs J_{i2}^A and J_i^B are processed in the same batch B_{i2} for $i \in S\backslash S'$. The constraint $C_{\max}^B \le Q_B = H^{2(m^2+m)+1}$ can be satisfied only if all the batches of $\{J_{i1}^A, J_i^B\}$ and $\{J_{i2}^A, J_i^B\}$ are first scheduled, so we have $\prod\limits_{i\in S'} a_i \le H$. The constraint $C_{\max}^A \le Q_A = H^{4(m^2+m)+1}$ can be satisfied only if $\prod\limits_{i\in S\backslash S'} a_i \le H$. Note that $\prod_{i\in S} a_i = D = H^2$, so we obtain $\prod\limits_{i\in S'} a_i = \prod\limits_{i\in S\backslash S'} a_i = H$, which gives a solution to the PP problem. $\Box$

4.1.2. $1|p-batch; p_j^X = \alpha_j^X(a+bt); c<n|\sum U_j^A : \sum U_j^B$

In this subsection, we discuss the complexity of the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c<n|\sum U_j^A : \sum U_j^B$. We will show that this scheduling problem is NP-hard for both incompatible and compatible cases by showing that the single-agent scheduling problem $1|p-batch; p_j = \alpha_j(a+bt); c<n|\sum U_j$ is NP-hard.

Theorem 4.1.3. The problem $1|p-batch; p_j = \alpha_j(a+bt); c<n|\sum U_j$ is NP-hard even if $c=2$.

Proof. We prove this by a reduction from the PP problem. Given an instance of PP, we construct an instance of the decision version of our problem as follows:

$n=4m$; $a=0$; $b=1$; $\alpha_{i1} = H^{4i}a_i - 1$, $\alpha_{i2} = \alpha_{i3} = \frac{H^{4i}}{a_i} - 1$; $d_{i1} = d_{i2} = d_{i3} = d_1 = H^{4(m^2+m)+1}$; $\alpha_{i4} = H^{4i} - 1$; $d_{i4} = d_2 = H^{2(m^2+m)+1}$; for $i = 1, 2, \ldots, m$; $t_0 = 1$; $c=2$; $Q=0$.

By similar arguments as in the proof of Theorem 4.1.2, we can show that there is a schedule for the instance with $\sum U_j \le Q$ if and only if there is a solution to the PP problem. We omit the details of the proof. $\square$

Similar to Theorem 3.2.5, we consider a special case of the problem $1|p-batch; p_j = \alpha_j(a+bt); c<n|C_{\max}^A : C_{\max}^B$ where the upper bound Q_B of the objective value $\sum U_j^B$ for agent B is sufficiently large. This case is equivalent to the single-agent problem $1|p-batch; p_j = \alpha_j(a+bt); c<n|\sum U_j$. Hence, we have the following result.

Theorem 4.1.4. The problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c<n|\sum U_j^A : \sum U_j^B$ is NP-hard for both incompatible and compatible cases.

4.2. The bounded parallel-batching with distinct release dates

4.2.1. $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c<n|C_{\max}^A : C_{\max}^B$

In this subsection, we first prove that the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c<n|C_{\max}^A : C_{\max}^B$ is NP-hard for both incompatible and compatible cases. Then we consider several polynomially solvable cases for incompatible and compatible agents.

Similar to Theorems 3.2.5 and 4.1.4, we can prove that the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c<n|C_{\max}^A : C_{\max}^B$ is equivalent to the single-agent problem $1|p-batch; p_j = \alpha_j(a+bt); r_j; c<n|C_{\max}$ when the upper bound Q_B of the objective value $C_{\max}^B$ for agent B is sufficiently large. While the problem $1|p-batch; p_j = \alpha_j(a+bt); r_j; c<n|C_{\max}$ is ordinary NP-hard when $a=0$ and $b=1$ (Li et al., 2011). Hence, we can easily have the following result.

Theorem 4.2.1. The problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c<n|C_{\max}^A : C_{\max}^B$ is NP-hard for both incompatible and compatible cases.

4.2.1.1. Scheduling with l distinct normal processing times for agent A and a common release date for agent B.

In this subsection, we present an optimal algorithm for the incompatible scheduling problem in which the jobs of agent A have $l(l \geq 2)$ distinct normal processing times and the jobs of agent B have a common release date $r^B > 0$, where l is a fixed positive integer. Let $\alpha_1, \alpha_2, \ldots, \alpha_l$ be the l distinct normal processing times of agent A. We call the jobs of agent A with normal processing times α_i as type i. Let m_i be the number of jobs of type i, then we have $\sum_{i=1}^{l} m_i = n_A$. For ease of exposition, we denote the jth job of type i as $J_{i,j}^A$ and its corresponding release date can be denoted by $r_{i,j}^A$ for $i = 1, ..., l$ and $j = 1, ..., m_i$. We can easily obtain the following properties:

Lemma 4.2.2. For the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c < n; IF|C_{\max}^A : C_{\max}^B$, in which the jobs of agent A have l distinct normal processing times and the jobs of agent B have a common release date $r^B > 0$, there is an optimal schedule such that all jobs of agent B are consecutively scheduled at or after time r^B, and they follow the FBLNPT rule, i.e., full batch longest normal processing time, and the jobs of the same type belonging to agent A are processed in non-decreasing order of their release dates.

Lemma 4.2.3. For the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c < n; IF|C_{\max}^A : C_{\max}^B$, in which the jobs of agent A have l distinct normal processing times and the jobs of agent B have a common release date $r^B > 0$, there is an optimal schedule for the jobs of agent A such that each batch is full with the possible exception of the first batch for the batches containing some jobs of the same type.

Algorithm DP6

According to Lemma 4.2.2, we first index the jobs of the same type of agent A such that $r_{i,1}^A \leq r_{i,2}^A \leq \cdots \leq r_{i,m_i}^A$ for $i = 1, ..., l$. Let $f(h_1, h_2, ..., h_l; n_1, n_2, ..., n_l)$ be the minimum completion time of A-agent's jobs that are processed before agent B satisfying the following conditions: (i) we have assigned jobs $J_{i,1}^A, J_{i,2}^A, ..., J_{i,h_i}^A$ for each type $i = 1, ..., l$ before agent B; (ii) the total number of jobs of type i processed before agent B is at most n_i for $i = 1, ..., l$ and $0 \leq h_i \leq n_i \leq m_i$; (iii) the last batch contains the last s_i jobs of type i (i.e. jobs $J_{i,h_i-s_i+1}^A, ..., J_{i,h_i}^A$) and $0 \leq s_i \leq h_i$ for $i = 1, ..., l$; (iv) the size of the last batch is not more than the capacity constraint, i.e., $\sum_{i=1}^{l} s_i \leq c$.

The initial condition is $f(0, 0, ..., 0; n_1, n_2, ..., n_l) = 0$ and

$$f(h_1,h_2,...,h_l;n_1,n_2,...,n_l)=\begin{cases}+\infty, & \text{if } s_i>\min\{c,h_i\},\\ +\infty, & \text{if } \sum_{i=1}^{l}s_i>\min\{c,\sum_{i=1}^{l}h_i\},\\ +\infty, & \text{if for some } i',\ \text{satisfy } 0<s_{i'}<h_{i'} \text{ and } \sum_{i=1}^{l}s_i<c.\end{cases}$$

The recursive relation is $f(h_1,h_2,...,h_l;n_1,n_2,...,n_l)=\min\{\max\{f(h_1-s_1,h_2-s_2,...,h_l-s_l;n_1,n_2,...,n_l),r(h_1,h_2,...,h_l;n_1,n_2,...,n_l)\}(1+b\alpha(h_1,h_2,...,h_l;n_1,n_2,...,n_l))+a\alpha(h_1,h_2,...,h_l;n_1,n_2,...,n_l):0\le s_i\le h_i\le n_i\le m_i,1\le i\le l,1\le\sum_{1\le i\le l}s_i\le c\}$, where $r(h_1,h_2,...,h_l;n_1,n_2,...,n_l)=\max\{r^A_{i,h_i}:s_i>0,1\le i\le l\}$ and $\alpha(h_1,h_2,...,h_l;n_1,n_2,...,n_l)=\max\{\alpha^A_i:s_i>0,1\le i\le l\}$ denote the release date and the normal processing time of the last batch, respectively.

The optimal solution value is $\min\{f(m_1,m_2,...,m_l;n_1,n_2,...,n_l)\}$ if $\max\{\min\{f(m_1,m_2,...,m_l;n_1,n_2,...,n_l)\},r^B\}\prod_{k=1}^{\lceil\frac{n_B}{c}\rceil}(1+b\alpha^B_k)+\frac{a}{b}(\prod_{k=1}^{\lceil\frac{n_B}{c}\rceil}(1+b\alpha^B_k)-1)\le Q_B$, where α^B_k is the normal processing time of each batch of agent B according to the FBLNPT rule.

If $\max\{\min\{f(m_1,m_2,...,m_l;n_1,n_2,...,n_l)\},r^B\}\prod_{k=1}^{\lceil\frac{n_B}{c}\rceil}(1+b\alpha^B_k)+\frac{a}{b}(\prod_{k=1}^{\lceil\frac{n_B}{c}\rceil}(1+b\alpha^B_k)-1)>Q_B$, then for each value $f(n_1,n_2,...,n_l;n_1,n_2,...,n_l)$, satisfying $\max\{\min\{f(n_1,n_2,...,n_l;n_1,n_2,...,n_l)\},r^B\}\prod_{k=1}^{\lceil\frac{n_B}{c}\rceil}(1+b\alpha^B_k)+\frac{a}{b}(\prod_{k=1}^{\lceil\frac{n_B}{c}\rceil}(1+b\alpha^B_k)-1)\le Q_B$, use the above recursive relation to compute the minimum completion time of A-agent's jobs that are processed after agent B, i.e., $f'(m_1-n_1,m_2-n_2,...,m_l-n_l;m_1-n_1,m_2-n_2,...,m_l-n_l)$. At this time, the initial condition is $f'(0,0,...,0;m_1-n_1,m_2-n_2,...,m_l-n_l)=\max\{f(n_1,n_2,...,n_l;n_1,n_2,...,n_l),r^B\}\prod_{k=1}^{\lceil\frac{n_B}{c}\rceil}(1+b\alpha^B_k)+\frac{a}{b}(\prod_{k=1}^{\lceil\frac{n_B}{c}\rceil}(1+b\alpha^B_k)-1)$. Hence, the optimal solution value is $\min\{f'(m_1-n_1,m_2-n_2,...,m_l-n_l;m_1-n_1,m_2-n_2,...,m_l-n_l)\}$.

It is clear that the complexity of the algorithm is $O(n_A^{2l}c^l)$.

4.2.1.2. Scheduling with agreeable (reversely agreeable) release dates and normal processing times.

In this subsection, we consider the compatible scheduling problem in which all job release dates and normal processing times are agreeable, i.e., $r_i<r_j$ implies $\alpha_i\le\alpha_j$, denoted by $agr(r_j,\alpha_j)$, or the job release dates and normal processing times are reversely agreeable, i.e., $r_i<r_j$ implies $\alpha_i\ge\alpha_j$, denoted by $revagr(r_j,\alpha_j)$. By using job-interchange argument, we have the following properties.

Lemma 4.2.4. For both of the problems $1|p-batch;p^X_j=\alpha^X_j(a+bt);agr(r_j,\alpha_j);c<n;CF|C^A_{\max}:C^B_{\max}$ and $1|p-batch;p^X_j=\alpha^X_j(a+bt);revagr(r_j,\alpha_j);c<n;CF|C^A_{\max}:C^B_{\max}$, there is an optimal schedule in the form (π_1,π_2,π_3) that has the following properties:

1) the partial schedule π_2 contains only all the *B-pure* batches, π_3 contains part of *A-pure* batches (if any), and π_1 contains the remaining batches;

2) all the jobs (batches) in the partial schedules π_1 and π_2 are scheduled in non-decreasing order of their release dates, all the jobs (batches) in the partial schedules π_1 and π_3 are also scheduled in non-decreasing order of their release dates.

Algorithm 2

By Lemma 4.2.4, we may assume that the jobs have been indexed as follows: $J_1^X, \ldots, J_{i-1}^X, J_i^A, J_{i+1}^B, \ldots, J_k^B, J_{k+1}^A, \ldots, J_n^A$, $X \in \{A, B\}$, such that $r_1 \leq r_2 \leq \cdots \leq r_n$. The definition of each parameter is the same as in Algorithm 1. Define $F(j)$ as the minimum completion time of a partial schedule containing jobs $J_1^X, \ldots, J_j^X$, for $X \in \{A, B\}$. Using the solution method for the single-agent problem $1|p-batch; p_j = \alpha_j(a+bt); r_j; c < n|C_{\max}$ that is similar to Algorithm DP^2 proposed by Li et al (2011), we can obtain the minimum completion time $F(j)$ of any one job J_j $j = 1, 2, \ldots, n$ for a given schedule sequence in $O(nc)$ time. The minimum completion time of job J_k^B in the partial schedule of jobs $J_1^X, \ldots, J_{i-1}^X, J_i^A, J_{i+1}^B, \ldots, J_k^B$ is denoted by $F_{opt}(k)$.

1) If $F_{opt}(k) \leq Q_B$, then we can compute $F(k+1), \ldots, F(n)$ for job sequence $J_1^X, \ldots, J_{i-1}^X, J_i^A, J_{i+1}^B, \ldots, J_k^B, J_{k+1}^A, \ldots, J_n^A$, $X \in \{A, B\}$. This takes $O(nc)$ time.

(1) We find the first job $J_j \in J^A$ for $k+1 \leq j \leq n$ such that $F(j) > Q_B$. We denote this job by J_{j^*}, i.e., $F(j^*) > Q_B$.

For each $j = j^*, j^*+1, \ldots, n$, if the last batch in the partial schedule of jobs $J_1^X, \ldots, J_{i-1}^X, J_i^A, J_{i+1}^B, \ldots, J_k^B, \ldots, J_j^A$ contains at least one job of agent B, then compute $F(j)$ as $F(j) = \min_{[j-(j^*-1)-c]^+ + j^*-1 \leq i \leq j-1} \{\max\{F(i), r_j\}(1 + b\alpha_j) + a\alpha_j\}$ (for the agreeable case) and $F(j) = \min_{[j-(j^*-1)-c]^+ + j^*-1 \leq i \leq j-1} \{\max\{F(i), r_j\}(1 + b\alpha_{i+1}) + a\alpha_{i+1}\}$ (for the reversely agreeable case), where $x^+ = \max\{x, 0\}$. There are at most n_A values for j and each value of j can be evaluated in $O(c)$ time. Hence, the running time for this case is $O(n_A c)$.

If the last batch in the partial schedule of jobs $J_1^X, \ldots, J_{i-1}^X, J_i^A, J_{i+1}^B, \ldots, J_k^B, \ldots, J_j^A$ does not contain any one job of agent B, then compute $F(j)$ as $F(j) = \min_{(j-c)^+ \leq i \leq j-1} \{\max\{F(i), r_j\}(1 + b\alpha_j) + a\alpha_j\}$ (for the agreeable case) and $F(j) = \min_{(j-c)^+ \leq i \leq j-1} \{\max\{F(i), r_j\}(1 + b\alpha_{i+1}) + a\alpha_{i+1}\}$ (for the reversely agreeable case). There are at most n_A possible values for j and each value of j can be evaluated in $O(c)$ time. Hence, the running time for this case is $O(n_A c)$.

(2) If $F(j) \leq Q_B$ for all $k+1 \leq j \leq n$, then the analysis is similar to (2) of Algorithm 1. The running time for this case is $O(n n_B c)$.

2) If $F_{opt}(k) > Q_B$, then the analysis is similar to 2) of Algorithm 1. The running time for this case is $O(n n_A c)$.

4.2.2. $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c<n|\sum U_j^A : \sum U_j^B$

By Theorem 4.1.4, the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); c<n|\sum U_j^A : \sum U_j^B$ is NP-hard, so when the jobs have different release dates, the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c<n|\sum U_j^A : \sum U_j^B$ is also NP-hard for both incompatible and compatible cases. We consider the complexity of two special cases in both incompatible and compatible cases, respectively. We will show that the problems with agreeable release dates and due dates in both cases are NP-hard. And the problems with agreeable release dates, due dates, and normal processing times in both cases are solvable in polynomial time.

4.2.2.1. Scheduling with agreeable release dates and due dates.

Theorem 4.2.5. The single-agent problem $1|p-batch; p_j = \alpha_j(a+bt); r_j; c<n|\sum U_j$ is NP-hard even if the release dates and due dates are agreeable.

Proof. We prove this by a reduction from the 4-Product problem, which is NP-complete in the strong sense (Kononov, 1996).

An instance of the 4-Product problem can be stated as follows:

4-Product (4-P) problem: Given positive rational numbers $a_1, a_2, \ldots, a_{4p}$ and H such that $H^{\frac{1}{5}} < a_i < H^{\frac{1}{3}}$ for $i = 1, 2, \ldots, 4p$ and $\prod_{i=1}^{4p} a_i = H^p$, does there exist a partition of the set $X = \{1, 2, \ldots, 4p\}$ into p disjoint subsets $X_1, X_2, \ldots, X_p$ such that $\prod_{i \in X_k} a_i = H$ for $k = 1, 2, \ldots, p$?

The decision version of the problem $1|p-batch; p_j = \alpha_j(a+bt); r_j; c<n|\sum U_j$ is clearly in NP. Given an instance of the 4-P problem, we construct an instance of the decision version of the single-agent problem as follows:

There are $n = 10p$ jobs. Let $a = 0$ and $b = 1$.

The normal processing times are defined by

$$\alpha_j = \begin{cases} H-1, & j = 1, \ldots, p, \\ a_{\lfloor \frac{1}{2}(j-p+1) \rfloor} - 1, & j = p+1, \ldots, 9p, \\ H-1, & j = 9p+1, \ldots, 10p, \end{cases}$$

The release dates are defined by

$$r_j = t_0 = 1, \text{ for } j = 1, \ldots, 9p; \; r_{9p+i} = H^{2i-1}, \text{ for } i = 1, \ldots, p.$$

The due dates are defined by

$$d_j = H^{2j}, \text{ for } j = 1, \ldots, p; \; d_j = H^{2p}, \text{ for } j = p+1, \ldots, 10p.$$

The capacity of bounded batch and the threshold value are defined by $c = 2$; $Q = 0$.

It can be seen that the above reduction from the strong NP-complete 4-P problem is polynomial with respect to the input problem length. But the magnitude of the resulting problem parameters is not bounded by a polynomial in the length and the magnitude of the 4-P problem, and so the reduction is not pseudo-polynomial. Consequently we are proving that $1|p-batch; p_j = \alpha_j(a+bt); r_j; c < n|\sum U_j$ is NP-hard in ordinary sense. We now show that there is a schedule to the instance with $\sum U_j \le Q$ if and only if there is a solution to the 4-P problem.

If Part. Suppose that there are disjoint subsets $X_1, X_2, \ldots, X_p$ with $X_k = \{l_{k1}, l_{k2}, \ldots, l_{k,n_k}\}$ and $\prod_{i\in X_k} a_i = H$ for $k = 1, 2, \ldots, p$, where $\sum\limits_{k=1}^{p} n_k = 4p$. For $i = 1, \ldots, 4p$, we put jobs J_{p+2i-1} and J_{p+2i} in a batch. Each batch corresponds to an element of the set X_k. This batch is denoted by $\{J_{p+2i-1}, J_{p+2i}\}$ and has normal processing time $a_i - 1$. For $i = 1, \ldots, p$, we put jobs J_i and J_{9p+i} in a batch. This batch is denoted by $\{J_i, J_{9p+i}\}$ and has normal processing time $H - 1$. We construct a batch sequence for the instance as follows:

$\{J_{p+2l_{11}-1}, J_{p+2l_{11}}\}$, $\{J_{p+2l_{12}-1}, J_{p+2l_{12}}\}$, $\ldots$, $\{J_{p+2l_{1,n_1}-1}, J_{p+2l_{1,n_1}}\}$, $\{J_1, J_{9p+1}\}$;

$\{J_{p+2l_{21}-1}, J_{p+2l_{21}}\}$, $\{J_{p+2l_{22}-1}, J_{p+2l_{22}}\}$, $\ldots$, $\{J_{p+2l_{2,n_2}-1}, J_{p+2l_{2,n_2}}\}$, $\{J_2, J_{9p+2}\}$;

$\cdots\cdots$

$\{J_{p+2l_{p1}-1}, J_{p+2l_{p1}}\}$, $\{J_{p+2l_{p2}-1}, J_{p+2l_{p2}}\}$, $\ldots$, $\{J_{p+2l_{p,n_p}-1}, J_{p+2l_{p,n_p}}\}$, $\{J_p, J_{10p}\}$.

It is easily verified that this schedule does not have any tardy jobs, i.e. $\sum U_j \le Q = 0$.

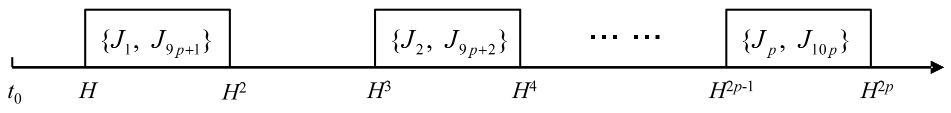

Figure 4. Jobs $J_i, J_{9p+i} (i = 1, \ldots, p)$ in the proof of Theorem 4.2.5.

Only If Part. Given a solution to the instance with $\sum U_j \le Q = 0$, we can conclude that there must be no idle time in this schedule and all batches must be full, while the two jobs in each batch must have identical normal processing times. This is because $(\prod\limits_{j=1}^{10p}(1+\alpha_j))^{\frac{1}{c}} = H^{2p} = \max\limits_j\{d_j\}$. In addition, for $i = 1, \ldots, p$, since $\alpha_i = \alpha_{9p+i} = H - 1$, $r_i = t_0 = 1$, $r_{9p+i} = H^{2i-1}$ and $d_i = H^{2i}$, jobs J_i and J_{9p+i} must form a batch with starting time H^{2i-1} and finishing time H^{2i} (see Figure 4). Thus, all the remaining jobs J_{p+j}, $j = 1, \ldots, 8p$, must fit into the time intervals $[H^{2k-2}, H^{2k-1}]$, $k = 1, 2, \ldots, p$. Since the two jobs in each batch must have identical normal processing times, we can construct batches as follows: jobs J_{p+2i-1} and J_{2i} form a batch, for $i = 1, \ldots, 4p$. Note that the number of tardy jobs is zero in such batch scheduling. Since

$\prod_{h=1}^{4p}(1+\alpha_h^A) = \prod_{h=1}^{4p} a_h = H^p$ and each of these intervals $[H^{2k-2}, H^{2k-1}]$ has a length of H time units, the sets of batches within these intervals corresponding to the sets J_{X_k} for $k = 1, 2, \ldots, p$. Hence $\prod_{i \in X_k} a_i = H$, which gives a solution to the **4-P** problem. □

By the preceding proof, we can conclude that when the upper bound value Q_B of agent B is sufficiently large, the two-agent scheduling problem is equivalent to the single-agent scheduling problem. Hence, we have the following result.

Theorem 4.2.6. The problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; c < n|\sum U_j^A : \sum U_j^B$ is NP-hard for both incompatible and compatible cases when the release dates and due dates are agreeable.

4.2.2.2. Scheduling with agreeable release dates, due dates and normal processing times.

We first present a polynomial time algorithm for the incompatible case in which the jobs of each agent have agreeable release dates, due dates and normal processing times, denoted by $agr(r_j^X, d_j^X, \alpha_j^X)$ for $X = A, B$. We assume that the jobs of each agent have been re-indexed such that $r_1^X \leq \cdots \leq r_{n_X}^X$, $d_1^X \leq \cdots \leq d_{n_X}^X$, and $\alpha_1^X \leq \cdots \leq \alpha_{n_X}^X$. We have the following property.

Lemma 4.2.7. For the problem $1|p-batch; p_j^X = \alpha_j^X(a+bt); r_j^X; agr(r_j^X, d_j^X, \alpha_j^X), X = A, B; c < n; IF|\sum U_j^A : \sum U_j^B$, there is an optimal schedule which has the form (E, L), where E is the set of batches containing all the early jobs and L is the set of batches containing all the late jobs. Moreover, all the early jobs that belong to each agent are scheduled in non-decreasing order of their indices and the early batches contain only consecutive jobs.

Algorithm DP7

Define $F(h_A, u_A, k_B, u_B)$ as the minimum completion time of the last early batch of a partial schedule of jobs $J_1^A, J_2^A, \ldots, J_{h_A}^A, J_1^B, J_2^B, \ldots, J_{k_B}^B$, where the numbers of early jobs of agents A and B are at least u_A and u_B, respectively. If there is no feasible schedule, we define $F(h_A, u_A, k_B, u_B) = +\infty$. The initial conditions are $F(0, 0, 0, 0) = 0$; $F(h_A, u_A, k_B, u_B) = +\infty$, if $h_A < 0$ or $k_B < 0$ or $u_A < 0$ or $u_B < 0$. By Lemma 4.2.7, we only consider the early jobs. When the last scheduled job belongs to agent A: if $J_{h_A}^A$ is tardy, then $F(h_A, u_A, k_B, u_B) = F(h_A - 1, u_A, k_B, u_B)$; if $J_{h_A}^A$ is early, then based on Lemma 4.2.7, the last early batch will contain l_A $(1 \leq l_A \leq b)$ consecutive jobs of agent A. When the last scheduled job belongs to agent B, we have a similar analysis. Hence, the recursive relation is

$$F(h_A,u_A,k_B,u_B)=\min\begin{cases}\min\{\min\limits_{1\le l_A\le \min\{c,u_A\}}\{F_{l_A}(h_A,u_A,k_B,u_B)\},F(h_A-1,u_A,k_B,u_B)\}\\ \qquad\qquad\text{if the last scheduled job belongs to agent } A,\\ \min\{\min\limits_{1\le l_B\le \min\{c,u_B\}}\{F_{l_B}(h_A,u_A,k_B,u_B)\},F(h_A,u_A,k_B-1,u_B)\}\\ \qquad\qquad\text{if the last scheduled job belongs to agent } B.\end{cases}$$

where

$$F_{l_A}(h_A,u_A,k_B,u_B)=\begin{cases}\max\{F(h_A-l_A,u_A-l_A,k_B,u_B),r^A_{h_A}\}(1+b\alpha^A_{h_A})+a\alpha^A_{h_A},\\ \quad\text{if } \max\{F(h_A-l_A,u_A-l_A,k_B,u_B),r^A_{h_A}\}(1+b\alpha^A_{h_A})+a\alpha^A_{h_A}\\ \qquad\qquad\le d^A_{h_A-l_A+1},\\ +\infty,\ \text{otherwise.}\end{cases}$$

$$F_{l_B}(h_A,u_A,k_B,u_B)=\begin{cases}\max\{F(h_A,u_A,k_B-l_B,u_B-l_B),r^B_{k_B}\}(1+b\alpha^B_{k_B})+a\alpha^B_{k_B},\\ \quad\text{if } \max\{F(h_A,u_A,k_B-l_B,u_B-l_B),r^B_{k_B}\}(1+b\alpha^B_{k_B})+a\alpha^B_{k_B}\\ \qquad\qquad\le d^B_{k_B-l_B+1},\\ +\infty,\ \text{otherwise.}\end{cases}$$

Here, $F_{l_A}(h_A,u_A,k_B,u_B)$ denotes the completion time of the last early batch of a partial schedule of jobs $J^A_1,J^A_2,\dots,J^A_{h_A},J^B_1,J^B_2,\dots,J^B_{k_B}$ when l_A jobs (i.e., $J^A_{h_A-l_A+1},J^A_{h_A-l_A+2},\dots,J^A_{h_A}$) of agent A are processed in the last early batch, and $F_{l_B}(h_A,u_A,k_B,u_B)$ denotes the completion time of the last early batch when l_B jobs (i.e., $J^B_{k_B-l_B+1},J^B_{k_B-l_B+2},\dots,J^B_{k_B}$) of agent B are processed in the last early batch.

The optimal solution value is $n_A-\max\{u_A|F(n_A,u_A,n_B,u_B)<+\infty,u_B\ge n_B-Q_B,1\le u_A\le n_A\}$. The complexity of this algorithm is $O(n_A^2n_B^2c)$.

In the following, we present a polynomial time algorithm for the compatible case in which the jobs have agreeable release dates, due dates, and normal processing times, denoted by $agr(r_j,d_j,\alpha_j)$. We may assume that the jobs have been re-indexed such that $r_1\le\cdots\le r_n$, $d_1\le\cdots\le d_n$, and $\alpha_1\le\cdots\le\alpha_n$. It is easy to see that Lemma 4.2.7 holds for all jobs in this problem as well.

Algorithm DP8

We may assume that the jobs are numbered from J_1 to J_n according to the EDD sequence. Define $F(j,u_A,u_B)$ as the minimum completion time of the last early batch of a partial schedule containing jobs $J_1, J_2, \ldots, J_j$, where the numbers of early jobs of agents A and B are at least u_A and u_B, respectively. The initial conditions are $F(0,0,0) = 0$; $F(j, u_A, u_B) = +\infty$, if $j < 0$ or $u_A < 0$ or $u_B < 0$. The recursive relation is given by

$$F(j, u_A, u_B) = \min\{\min_{1 \le l \le \min\{c, u_A + u_B\}} \{F_l(j, u_A, u_B)\}, F(j-1, u_A, u_B)\},$$

where for $l_A + l_B = l$,

$$F_l(j, u_A, u_B) = \begin{cases} \max\{F(j-l, u_A - l_A, u_B - l_B), r_j\}(1 + b\alpha_j) + a\alpha_j, \\ \qquad \text{if } \max\{F(j-l, u_A - l_A, u_B - l_B), r_j\}(1 + b\alpha_j) + a\alpha_j \le d_{j-l+1}, \\ +\infty, \quad \text{otherwise.} \end{cases}$$

Here, $F_l(j, u_A, u_B)$ denotes the completion time of the last early batch of a partial schedule containing jobs $J_1, J_2, \ldots, J_j$, when l jobs (i.e., $J_{j-l+1}, J_{j-l+2}, \ldots, J_j$) are processed in the last early batch, in which l_A jobs are from agent A and l_B jobs are from agent B, i.e., $l_A + l_B = l$. The optimal solution value is $n_A - \max\{u_A | F(j, u_A, u_B) < +\infty, u_B \ge n_B - Q_B, 1 \le u_A \le n_A\}$. The complexity of this algorithm is $O(n n_A n_B c)$.

5. Conclusions

In this paper, we studied several two-agent scheduling problems on a batching machine with time-dependent proportional-linear deteriorating job processing times. We focused on minimizing the objective of one agent subject to an upper bound on the objective of the other agent. The objective functions considered include the number of tardy jobs, the makespan and the maximum of regular functions of job completion times. We provided either polynomial time algorithms or NP-hardness proofs for general problems, in which the jobs may have the same or different release dates for incompatible and compatible cases. We also provided polynomial or pseudo-polynomial time algorithms for certain special cases of the intractable problems. For future research, it will be interesting to develop exact or approximate solution algorithms for the intractable problems. Future research may also consider other objective functions, such as the total completion time.

Acknowledgment

This research is partly supported by State Key Program of National Natural Science Foundation of China (71032004).

References

Agnetis, A., Billaut, J.C, Gawiejnowicz, S., Pacciarelli, D., & Soukhal, A. (2014). Multiagent scheduling: models and algorithms. Berlin, Heidelberg: Springer.

Agnetis, A., Mirchandani, P.B., Pacciarelli, D., & Pacifici, A. (2004). Scheduling problems with two competing agents. *Operations Research, 52*, 229–242.

Baker, K.R., & Smith, J.C. (2003). A multiple-criterion model for machine scheduling. *Journal of Scheduling, 6*, 7–16.

Cheng, T.C.E., Ding, Q., & Lin, B.M.T. (2004). A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research, 152*, 1–13.

Fan, B.Q., Cheng, T.C.E., Li, S.S., & Feng, Q. (2013). Bounded parallel-batching scheduling with two competing agents. *Journal of Scheduling, 16*, 261–271.

Gawiejnowicz, S. (2008). Time-Dependent Scheduling. Springer, Berlin.

Gawiejnowicz, S., Lee, W.C., Lin, C.L., & Wu, C.C. (2011). Single-machine scheduling of proportionally deteriorating jobs by two agents. *Journal of the Operational Research Society, 62*, 1983–1991.

Gawiejnowicz, S., & Suwalski, C. (2014). Scheduling linearly deteriorating jobs by two agents to minimize the weighted sum of two criteria. *Computers & Operations Research, 52*, 135–146.

He, C., & Leung J.Y.T. (2016). Two-agent scheduling of time-dependent jobs. *Journal of Combinatorial Optimization*, available online, DOI: 10.1007/s10878-016-9994-y.

Kononov, A. (1996). Combinatorial complexity of scheduling jobs with simple linear processing times. *Diskretny Analiz i Issledovanie Operatsii, 3*, 15–32 (in Russian).

Lee, K.H. (1979). Dynamic scheduling of ingot processing in steel production. Ph.D. Thesis, Purdue University.

Leung, J.Y.T., Pinedo, M.L., & Wan, G. (2010). Competitive two agent scheduling and its applications. *Operations Research, 58*, 458–469.

Li, S.S., Ng, C.T., Cheng, T.C.E., & Yuan, J.J. (2011). Parallel-batch scheduling of deteriorating jobs with release date to minimize the makespan. *European Journal of Operational Research, 210*, 482–488.

Li, S.S., & Yuan, J.J. (2012). Unbounded parallel-batching scheduling with two competitive agents. *Journal of Scheduling, 15*, 629–640.

Liu, P., & Tang, L.X. (2008). Two-agent scheduling with linear deteriorating jobs on a single machine. *Lecture Notes in Computer Science, 5092*, 642–650.

Liu, P., Tang, L.X., & Zhou, X.Y. (2010). Two-agent group scheduling with deteriorating jobs on a single machine. *The International Journal of Advanced Manufacturing Technology, 47*, 657–664.

Liu, P., Yi, N., & Zhou, X.Y. (2011). Two-agent single-machine scheduling problems under increasing linear deterioration. *Applied Mathematical Modelling, 35*, 2290–2296.

Miao, C.X., Zhang, Y.Z., & Wu, C.L. (2012). Scheduling of deteriorating jobs with release dates to minimize the maximum lateness. *Theoretical Computer Science, 462*, 80–87.

Ng, C.T., Barketau, M.S., Cheng, T.C.E., & Kovalyov, M.Y. (2010). "Product Partition" and related problems of scheduling and systems reliability: Computational complexity and approximation. *European Journal of Operational Research, 207*, 601–604.

Patel, C., Ray, W.H., & Szekely, J. (1976). Computer simulation and optimal scheduling of a soaking pit-slabbing mill system. *Metallurgical Transactions B, 7B*, 119–130.

Perez-Gonzalez, P., & Framinan, J.M. (2014). A common framework and taxonomy for multicriteria scheduling problem with interfering and competing jobs: Multi-agent scheduling problems. *European Journal of Operational Research, 235*, 1–16.

Potts, C.N., & Kovalyov, M.Y. (2000). Scheduling with batching: a review. *European Journal of Operational Research, 120*, 228–249.

Yin, Y., Cheng, T.C.E., Wan, L., Wu, C.C., & Liu, J. (2015). Two-agent single-machine scheduling with deteriorating jobs. *Computers & Industrial Engineering, 81*, 177–185.

Yuan, J.J., Qi, X.L., Lu, L.F., & Li, W.H. (2008). Single machine unbounded parallel-batch scheduling with forbidden intervals. *European Journal of Operational Research, 186*, 1212–1217.