Hindawi
Complexity
Volume 2020, Article ID 6840471, 10 pages
https://doi.org/10.1155/2020/6840471

Research Article

Scheduling on a Single Machine and Parallel Machines with Batch Deliveries and Potential Disruption

Hua Gong,[1,2] **Yuyan Zhang**,[1] **and Puyu Yuan**[1]

[1]*College of Science, Shenyang Ligong University, Shenyang 110159, China*
[2]*Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, Shenyang 110819, China*

Correspondence should be addressed to Hua Gong; gonghua@sylu.edu.cn and Puyu Yuan; 452983100@qq.com

Received 8 May 2020; Revised 30 June 2020; Accepted 20 July 2020; Published 14 September 2020

Guest Editor: Baogui Xin

In this paper, we study several coordinated production-delivery scheduling problems with potential disruption motivated by a supply chain in the manufacturing industry. Both single-machine environment and identical parallel-machine environment are considered in the production part. The jobs finished on the machines are delivered to the same customer in batches. Each delivery batch has a capacity and incurs a delivery cost. There is a situation that a possible disruption in the production part may occur at some particular time and will last for a period of time with a probability. We consider both resumable case and nonresumable case where a job does not need (needs) to restart if it is disrupted for a resumable (nonresumable) case. The objective is to find a coordinated schedule of production and delivery that minimizes the expected total flow times plus the delivery costs. We first present some properties and analyze the NP-hard complexity for four various problems. For the corresponding single-machine and parallel-machine scheduling problems, pseudo-polynomial-time algorithms and fully polynomial-time approximation schemes (FPTASs) are presented in this paper, respectively.

1. Introduction

Production scheduling and delivery decision are two key operations in the supply chain management. Coordinated production and delivery schedules play an important role in improving optimal operational performance in the supply chain of high energy-consuming manufacturing industry. However, traditionally, the scheduling decisions of coordinated production and delivery in a firm are often made in a certain environment. In a real industry, the production scheduling environment is dynamic and uncertain. Unexpected events may occur from some time point and will last a period. There is a situation that the internal and external factors cause various disruptions since the machines or facilities are disrupted and unavailable for a certain period of time during production, such as material shortages, machine breakdowns, power failures, quality issues, and others. This will lead to affect production efficiency, extra energy consumption, and logistics management in the service system.

For example, in the iron and steel industry, the slabs are first rolled into the coils in the hot rolling mill. The coils are transported by the trucks to the customers or the downstream facilities for further processing. The slabs need high temperature to ensure to be processed on the hot rolling machines. If the disruption occurs on the hot rolling machines, it will lead to the temperature decrease of the slabs. When the disruption time exceeds a limit, the slabs must to return to the heating furnaces to re-heat to keep high temperature. This disruption will not only affect the production rhythm and increase the energy consumption in the mill but also influence the delivery decision for the customers. Hence, the reasonable scheduling for the disruption affects the revenue in the iron and steel industry. The inventory level and the total delivery cost are the major concerns for the industry managers. The inventory level is measured by a function of the flow time when the finished coils arrive to the customers. It is well understood that coordinated scheduling of production and delivery can

significantly improve inventory level and reduce delivery cost.

In this paper, we study several coordinated production-delivery scheduling problems with potential disruption motivated by the dynamic and uncertain environment. All the jobs are first finished processing on the machines and then delivered to the customers in batches. A possible disruption in the production part may occur at a particular time and will last for a period of time with a probability. We investigate the optimal strategies on how to deal with disruptions in scheduling in the single-machine environment and identical parallel-machine environment, respectively. The objective is to find a feasible schedule to minimize the expected total flow times and the delivery costs.

Many coordinated scheduling problems with production and delivery in most manufacturing systems consider the balance relationship between the finished products and the transportation or delivery. Chen [1] presented a survey of integrated production and outbound distribution scheduling with batch delivery. Wang et al. [2] provided a survey of the integrated scheduling models of production and distribution. The coordinated scheduling problems of production and transportation with batching decisions in the iron and steel industry are considered in [3–6]. Some researchers addressed production and distribution scheduling of minimizing the sum of the total (weighted) flow time and delivery cost from a batch delivery point of view. This environment is related to batching because all the completed jobs are delivered in batches to the customer. Hall and Potts [7] considered the single-machine or identical parallel-machine scheduling problems in supply chain with batch deliveries. Wang and Cheng [8] considered a parallel-machine scheduling problem with batch delivery cost where the batch delivery date is equal to the completion time of the last job in a batch. Chen and Vairaktarakis [9] considered the single-machine and parallel-machine scheduling problems with distribution scheduling and routing for delivery of completed jobs to the customers. Wan and Zhang [10] studied the extension of the parallel-machine scheduling problem with batch delivery. The above scheduling literatures do not deal with the coordination of production and delivery related to potential disruptions.

The research on machine scheduling with potential disruption has some results in a significant amount of interest in production scheduling research. Lee and Yu [11] considered the single-machine scheduling problems with potential disruptions due to external factors where the objective functions are the expected total weighted completion times and the expected maximum tardiness. Lee and Yu [12] further considered the parallel-machine problems to minimize the expected total weighted completion time. Yin et al. [13] considered the parallel-machine scheduling problems with deterioration in a disruptive environment in which the machines may be unavailable due to potential disruptions. Zheng and Fan [14] studied the parallel-machine scheduling problems with position-dependent processing times under potential disruption. Lee et al. [15] considered a two-machine scheduling problem with disruptions and transportation considerations. In [15], once one machine undergoes disruption and is unavailable, the jobs can either be moved to other available machines for processing, which consider transportation time and transportation cost, or be processed by the same machine after the disruption.

In this paper, we study coordinated scheduling problems with potential disruption and batch deliveries considering single machine or identical parallel machines. Our problem is an extension of the problem proposed by Gong et al. [6] and Hall and Potts [7], who studied the single-machine and parallel-machine scheduling problems with batch delivery without potential disruption. On the other hand, our problem is a natural extension of the problem proposed by Lee and Yu [12], who studied the parallel-machine problem under potential disruption without production and delivery coordination. We develop coordinated scheduling on the single machine and parallel machines with both batch deliveries and potential disruption in this paper. Both resumable case and nonresumable case are considered in this paper where a job does not need (needs) to restart if it is disrupted for a resumable (nonresumable) case. The objective is to minimize the sum of the expected total flow time and total delivery costs. To the best of our knowledge, the scheduling problems with batch deliveries and potential disruption on the single machine and parallel machines have never been discussed. In this paper, we first pinpoint the difficulty by reducing the problems to be NP-hard. We also present pseudo-polynomial-time algorithms to solve these problems and further show that the problems proposed in this paper are NP-hard in the ordinary sense, respectively. Finally, we provide fully polynomial-time approximation schemes via the scaling and trimming techniques to solve the problems.

For many NP-hard scheduling problems, it will be clearly hard to find optimal schedules in a time-effective manner. The existence of a pseudo-polynomial-time algorithm for a NP-hard problem means that this NP-hard problem is ordinarily NP-hard, but not strongly NP-hard [16]. A fully polynomial-time approximation scheme (FPTAS) for a problem is an approximation scheme with a time complexity that is polynomial in the input size n and in $1/\varepsilon$ for any given $\varepsilon > 0$. With regard to worst-case approximations, an FPTAS is the strongest possible polynomial-time approximation result that one can obtain for an NP-hard problem unless $P = NP$ [17].

The rest of the paper is organized as follows. We introduce the problem description and some properties in Section 2. In Section 3, we provide the algorithms and approximation schemes for the single-machine scheduling problems. Section 4 deals with two-parallel-machine problems and develops the corresponding algorithms and approximation schemes. Section 5 provides a conclusion and some suggestions for future research.

2. Problem Description and Notation

In this section, we describe our problems briefly. There are n jobs $\{J_1, \ldots, J_n\}$ to be first processed on either a single machine or m identical parallel machines $M_1, \ldots, M_m$, all to

be delivered in batches to the same customer. Each machine can process at most one job at a time. All the jobs are available for processing at initial time. For job J_j, let a_j and C_j be the processing time and the completion time on the machine, respectively. Preemption may happen for some jobs if potential disruption of some machines happens. We assume that a machine disruption occurs at time point r and will last for α time units $(\alpha = 0, 1, \ldots, \gamma)$ with a certain probability p_α. The machines are the same for all the disrupted machines once the disruption occurs. Here, p_0 is the probability that the potential disruption will not happen and the machine is always available. Without loss of generality, we assume the machine disruption situation that the disruption may happen simultaneously on the first k machines $M_1, \ldots, M_k$, with $1 \le k \le m$. Let $A_j = \sum_{i=1}^{j} a_i$ represent the total processing time for the first jobs $J_1, \ldots, J_j$. Set $a_{\max} = \max\{a_j : j = 1, \ldots, n\}$.

Once the disruption machines have unavailable time intervals $[r, r + \alpha]$, the disrupted jobs need to restart on the machines. We consider two cases: resumable and nonresumable. If a job is disrupted during processing on some disrupted machine, it is called resumable (nonresumable) that the job does not need (needs) to restart when the machine becomes available again. For the resumable case, the remaining part of the disrupted job will continue at time point $r + \alpha$ without any penalty. For the nonresumable case, the disrupted job needs to restart at time point $r + \alpha$.

In the delivery part, a fleet of transporter for delivering the jobs in batches to the same customer will be viewed as a single transporter because they have common departure times and return times. A group of jobs forms a delivery batch B_l if all of these jobs are delivered to the customer simultaneously. The number of jobs in a delivery batch cannot exceed the delivery capacity C. Let y be the number of delivery batches for all jobs in a schedule. Delivering a batch will incur a fixed delivery cost D. The single transporter is located at the production area at initial time. Assume that $T/2$ is the transportation time from the machine to the customers, and the return time from the customer to the machine is $T/2$. There is at least minimum time interval T between any consecutive batches. Let F_j denote the flow time of job J_j when it is delivered to the customer. The objective that we consider is to schedule the jobs for production and delivery such that the sum of the expected total flow times $E[\sum F_j]$ and the delivery costs Dy is minimized. The problems considered in this paper can be expressed as $1|T|$, disruption$|E[\sum F_j] + Dy$ and $P_m|T|$, disruption$|E[\sum F_j] + Dy$ by adopting the notation introduced by Hall and Potts [7]. The four various problems can be expressed as follows:

$$\begin{aligned}
&P1: 1|T, \text{non-resumable, disruption}|E\left[\sum F_j\right] + Dy,\\
&P2: 1|T, \text{resumable, disruption}|E\left[\sum F_j\right] + Dy,\\
&P3: Pm|T, \text{non-resumable, disruption}|E\left[\sum F_j\right] + Dy,\\
&P4: Pm|T, \text{resumable, disruption}|E\left[\sum F_j\right] + Dy.
\end{aligned} \tag{1}$$

Next, we analyze the complexity of the problems studied in this paper.

For the single-machine scheduling problem with batch deliveries without delivery capacity, problem $1|T|\sum F_j + Dy$ proposed by Hall and Potts [7] is optimal to sequencing the jobs in SPT rule. Note that a special case of our problem $1|T$, disruption$|E[\sum F_j] + Dy$, in which the delivery part is ignored and the objective is to minimize $E[\sum C_j]$, was studied by Lee and Yu [11]. On the other hand, even if $p_0 = 1$ (it means that the disruption will certainly happen in a fixed period), problem 1|disruption$|E[\sum C_j]$ in [11] is ordinarily NP-hard. We extend $1|T|\sum F_j + Dy$ to consider disruption constraint and 1|disruption$|E[\sum C_j]$ to consider delivery coordination. Hence, $P1$ and $P2$ proposed in this paper are NP-hard.

For the parallel-machine scheduling problem with batch deliveries, $Pm|T|\sum F_j + Dy$ is shown NP-hard in the ordinary sense by Gong et al. [6] even if the potential disruption does not happen. Hence, $Pm|T$, disruption $|E[\sum F_j] + Dy$ is at least NP-hard even if $p_0 = 0$. If the delivery is ignored, two-parallel-machine scheduling problem with potential disruption is already NP-hard even if $p_0 = 1$ as presented by Lee and Yu [12]. Hence, our problems $P3$ and $P4$ are also NP-hard.

In [6, 7], the goal is to determine the job sequencing on the machines and the job partitioning into delivery batches, but not considering the machine potential disruption. In [11, 12], they made the production scheduling decision on potential disruption according to two sequential decisions at 0 and disruption r. In this paper, we need to not only sequence the jobs on the single machine or parallel machines and reschedule the jobs with disruption consideration in the production part but also partition the jobs into delivery batches in the delivery part. In this paper, we will derive the corresponding pseudo-polynomial-time algorithms for problem $1|T$, disruption$|E[\sum F_j] + Dy$ and $P_m|T$, disruption$|E[\sum F_j] + Dy$ based on dynamic programming, respectively. Furthermore, the fully polynomial-time approximation schemes are provided for four problems that show that the problems are NP-hard in the ordinary sense.

The following properties hold for both resumable and nonresumable cases.

2.1. General Results

Lemma 1. *For P1–P4, there exists an optimal schedule without idle time between jobs for any machine except during the machine disruption time interval.*

Lemma 2. *For P1–P4, there exists an optimal schedule that all the departure times of delivery batches are made either at the completion times of jobs or at immediate available times of the transporter.*

Lemma 3. *For P1 and P2, there exists an optimal schedule that the jobs finished no later than r are sequenced in the shortest processing time (SPT) rule, and the remaining jobs finished after r are also sequenced in the SPT rule.*

Proof. Assume an optimal schedule π^*. If π^* contains jobs that are not sequenced in SPT rule, then J_j is followed by J_i where $a_j > a_i$.

(1) If J_j and J_i are assigned into the same delivery batch, then the jobs can be sequenced in the SPT rule without affecting the objective cost.

(2) If J_j is the last job in batch B_{l-1} and J_i is the first job in batch B_l, then form a schedule π with $B_{l-1} \cup \{i\}\backslash\{j\}$ and $B_l \cup \{j\}\backslash\{i\}$ by interchanging J_j and J_i. Before r or after r, $B_{l-1} \cup \{i\}\backslash\{j\}$ in schedule π is delivered at the same time as batch B_{l-1} in schedule π^*, and $B_l \cup \{j\}\backslash\{i\}$ in schedule π is delivered at the same time as batch B_l in schedule π^*. All other delivery batches are identical and delivered at the same time in π as in π^*. Hence, the objective cost associated with π is the same as the objective cost associated with π^*. Repeating the argument can find an optimal schedule with SPT rule.

Lemma 4. *For P3 and P4, there exists an optimal schedule in which*

(1) The jobs finished no later than r are sorted in the SPT rule on each disrupted machine M_i, $i = 1, \ldots, k$, and sorted in the SPT rule on each nondisrupted machine M_i, $i = k+1, \ldots, m$, respectively.

(2) The remaining jobs finished after r are also sorted in the SPT rule on each disrupted machine M_i, $i = 1, \ldots, k$.

Proof. For any given scenario α, Lemma 3 shows that the subproblem is optimal to schedule the jobs finished no later than r or after r in the SPT rule when M_i can be viewed as a single machine. For all possible scenarios α and all the machines, this argument holds when the objective is to minimize $E[\sum F_j] + D_y$.

3. Single-Machine Scheduling

In this section, two pseudo-polynomial-time algorithms to solve two single-machine problems with the nonresumable case ($P1$) and the resumable case ($P2$) are developed, respectively. Furthermore, we will convert the dynamic programming algorithms into fully polynomial-time approximation schemes (FPTASs) to solve problems $P1$ and $P2$. We will recall the definition of an FPTAS in Section 3.3.

3.1. The Nonresumable Case P1. In this section, a pseudo-polynomial-time algorithm based on dynamic programming is presented to solve $P1$. At first, re-index jobs in the SPT rule. Fix the following variables: jobs $J_1, \ldots, J_j$ are assigned to two sets, S and $\{J_1, \ldots, J_j\}\backslash S$, such that $\sum_{i \in S} a_i = q \le r$ and the jobs in S will be processed consecutively without idle time on the single machine from initial time 0. Define t as the starting time of the first job in $\{J_1, \ldots, J_j\}\backslash S$ if the disruption does not happen but the first job will finish after time r. It is clear that t ranges from $\max\{q, r - a_{\max} + 1\}$ to r. Let $F(j, q, t, |B_l|)$ be the expected objective value of a feasible partial schedule for jobs $J_1, \ldots, J_j$, where $|B_l|$ is the number of jobs assigned into delivery batch B_l and the variable q denotes the total actual processing time of jobs in S finished no later than r on the machine. Let $F_{[l]}$ be the delivery completion time of batch B_l, which is the flow time of each job in B_l delivered to the customer. We provide a dynamic programming to solve problem $1|T$, nonresumable, disruption$|\ E[\sum F_j] + Dy$.

Theorem 1. *Algorithm 1 (DP1) can solve problem P1 in time complexity $O(n \log_n + r^2 n^2)$.*

Proof. We first analyze the possible recursive situations. Suppose that jobs $J_1, \ldots, J_{j-1}$ have been assigned optimally, and start to assign J_j. In an optimal schedule, J_j will be either the last job processed no later than t or last in the whole sequence. If J_j is assigned no later than t, then it will be completed at least at time q on the machine. On the other hand, if job J_j is assigned last in the whole sequence, then its completion time on the machine is at least $p_0(t + A_j - q) + \sum_{\alpha=1}^{s} p_\alpha (r + \alpha + A_j - q)$. Simultaneously, delivery batch B_l at either the completion time of job J_j or the return time of the transporter which has finished the previous batch to the machine. If J_i is assigned into a new batch, then the delivery cost contributes D. Hence, the objective contribution of a new batch is $F_{[l]} + D$. If J_i is assigned into the current batch, the objective contribution is $F_{[l]}$ and the delivery cost does not change. Simultaneously, the number of jobs in each delivery batch cannot exceed the delivery capacity C. The SPT rule needs $O(n \log_n)$ time to sequence the jobs in Step 1. By the definition of the recursive relations, we have $j, l \le n$ and $q, t \le r$. Hence, the overall complexity is $O(\mathrm{n} \log n + r^2 n^2)$.

3.2. The Resumable Case P2. Actually, Algorithm 1 (DP1) can be modified as below to solve problem $P2$ with resumable consideration.

Theorem 2. *Algorithm 2 (DP2) can solve problem P2 in time complexity $O(n \log n + r^2 n^2)$.*

Theorem 3. *P1 and P2 are NP-hard in the ordinary sense.*

3.3. Fully Polynomial-Time Approximation Schemes for P1 and P2. A fully polynomial-time approximation scheme for the single-machine scheduling problems is provided in this section. Let f^* represent the objective value of the optimal schedule and f represent the objective value of the schedule generated by the approximation algorithms. Note that a polynomial-time approximation scheme (PTAS) for a problem is a family of polynomial time $(1 + \varepsilon)$ – approximation algorithms if $f \le (1 + \varepsilon) f^*$, where $\varepsilon > 0$ is a bound on relative error of the algorithms. Furthermore, a special PTAS is called a fully polynomial-time approximation scheme (FPTAS) if its time complexity is polynomial in $1/\varepsilon$. In pure technical sense, an FPTAS is a best one that may tackle to solve an NP-hard optimization problem, unless $P = NP$ [16].

Step 1. Re-index the jobs in the SPT rule.
Step 2. Initial conditions: $F(0, 0, t, 0) = 0$, for $t = \max\{q, r - a_{\max} + 1\}, \ldots, r$.
Step 3. Recursive equations:
For $t = \max\{q, r - a_{\max} + 1\}, \ldots, r$, and $j = 1, \ldots, n$;
For $l = 1$ and $j \le C$, set $F_{[1]} = C_j + T/2$ and $F(j, q, t, |B_1|) = j(C_j + T/2) + D$.
For $l = 2$ to n do
If $t + A_j - q \le r$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), q\} + (T/2)$
When $|B_{l-1}| = C$, set $F(j, q, t, |B_l|) \longleftarrow F(j-1, q - a_j, t, |B_{l-1}|) + F_{[l]} + D$;
When $|B_{l-1}| < C$, set $F(j, q, t, |B_l|) \longleftarrow \min\{F(j-1, q - a_j, t, |B_{l-1}|) + F_{[l]} + D, F(j-1, q - a_j, t, |B_l|) + F_{[l]}\}$
Endif
If $t + A_j - q > r$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), p_0(t + A_j - q) + \sum_{\alpha=1}^{s} p_\alpha (r + \alpha + A_j - q)\} + (T/2)$
When $|B_{l-1}| = C$, set $F(j, q, t, |B_l|) \longleftarrow F(j-1, q, t, |B_{l-1}|) + F_{[l]} + D$;
When $|B_{l-1}| < C$, set $F(j, q, t, |B_l|) \longleftarrow \min\{F(j-1, q, t, |B_{l-1}|) + F_{[l]} + D, F(j-1, q, t, |B_l|)F_{[l]}\}$
Endfor
Endfor
Step 4. Optimal solution:
$\min\{F(j, q, t, |B_l|) : q = 0, \ldots, r; t = \max\{0, r - a_{\max} + 1\}, \ldots, r; \lceil n/C \rceil \le l \le n\}$.

Algorithm 1: Algorithm DP1.

Step 1. Perform Step 1-2 of Algorithm 1 (DP1).
Step 2. Recursive equations:
For $t = \max\{q, r - a_{\max} + 1\}, \ldots, r$, $j = 1, \ldots, n$;
For $l = 1$, i.e., $j \le C$, set $F_{[1]} = C_j + (T/2)$ and $F(j, q, t, |B_1|) = j(C_j + (T/2)) + D$.
For $l = 2$ to n do
If $t + A_j - q \le r$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), q\} + (T/2)$
When $|B_{l-1}| = C$, set $F(j, q, t, |B_l|) \longleftarrow F(j-1, q - a_j, t, |B_{l-1}|) + F_{[l]} + D$
When $|B_{l-1}| < C$, set $F(j, q, t, |B_l|) \longleftarrow \min\{F(j-1, q - a_j, t, |B_{l-1}|) + F_{[l]} + D, F(j-1, q - a_j, t, |B_l|)F_{[l]}\}$
Endif
If $t + A_j - q > r$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), \sum_{\alpha=0}^{s} p_\alpha (t + \alpha + A_j - q)\} + (T/2)$
When $|B_{l-1}| = C$, set $F(j, q, t, |B_l|) \longleftarrow F(j-1, q, t, |B_{l-1}|) + F_{[l]} + D$
When $|B_{l-1}| < C$, set $F(j, q, t, |B_l|) \longleftarrow \min\{F(j-1, q, t, |B_{l-1}|) + F_{[l]} + D, F(j-1, q, t, |B_l|) + F_{[l]}\}$
Endfor
Endfor
Step 3. Optimal solution:
$\min\{F(j, q, t, |B_l|) : q = 0, \ldots, r; t = \max\{0, r - a_{\max} + 1\}, \ldots, r; \lceil n/C \rceil \le l \le n\}$.

Algorithm 2: Algorithm DP2.

In the following, we present the general outline of an FPTAS for solving $P1$ and $P2$. This is done by applying the well-known scaling technique to formulate a certain scaled problem for the original problem. The pseudo-polynomial-time algorithm based on dynamic programming for the original problem can generate this scaled problem with its parameter. This algorithm to solve this scaled problem provides a fully polynomial-time approximation scheme for the original problem.

Next, we convert the pseudo-polynomial-time algorithm using the scaling technique into an FPTAS.

3.3.1. $FPTAS_S$ for $P1$ and $P2$

Step 1. Given an arbitrary $\varepsilon > 0$, we define $\delta = \varepsilon r / n(2n\gamma + 3)$.

Step 2. Construct a scaled problem. Define new parameters of each job J_j for the scaled problem:

$$a_j' = \lfloor \frac{a_j}{\delta} \rfloor, \quad T' = \lfloor \frac{T}{\delta} \rfloor, \quad r' = \lfloor \frac{r}{\delta} \rfloor, \quad D' = \lfloor \frac{D}{\delta} \rfloor, \quad j = 1, 2, \ldots, n. \tag{2}$$

Step 3. For the scaled problem, apply Algorithms 1 (DP1) and 2 (DP2) to obtain a schedule $\pi\prime$ with minimal objective value.

Let $\pi\prime$ be an optimal schedule and F' be its objective value for the scaled problem. Suppose that π^* for the

original problem is an optimal schedule and its objective value is F^*. Note that the schedule has at most $n\gamma$ processing operations if the potential disruption occurs. For each state $(i, q, t|B_l|)$, increase each processing time by $a_j/\delta - a_j'$ before disruption q or after disruption, which increases C_j by at most $2n\gamma$. Increase each disruption time point by $r/\delta - r'$ and each returning time by $T/\delta - T'$. Increase each delivery cost by $D/\delta - D'$. Further, the total flow time of all n jobs increases at most $n(2n\gamma + 3)$. Now consider $1/\delta$ to be a unit. Then, we obtain a schedule $\pi\prime$ with an approximate solution for the original problem, and its objective value can be formulated as

$$\frac{F(\pi')}{\delta \le F'(\pi')} + n(2n\gamma + 3). \tag{3}$$

We have $F'(\pi') \le F'(\pi^*)$ due to the definition of π'. It follows that $F'(\pi^*) \le F^*(\pi^*)/\delta$ from $\lfloor x \rfloor \le x$ for any number x. We obtain

$$F(\pi') \le \delta F(\pi^*) + n\delta(2n\gamma + 3) \le F^*(\pi^*) + \varepsilon r \le (1+\varepsilon)F^*(\pi^*). \tag{4}$$

Thus, we can see that the approximate solution is at most a factor of $1 + \varepsilon$ away from the optimum solution.

The time complexity of the approximation scheme is dominated by the step to solve the scaled problem. It is easy to see that $r' \le (r/\delta) = (n(2n\gamma + 3)/\varepsilon)$. Thus, the running time of the approximation scheme is bounded by

$$\begin{aligned} O\left(n^2(r')^2 + n\log n\right) &\le O\left(n^2\left(\frac{n(2n\gamma+3)}{\varepsilon}\right)^2\right) \\ &\le O\left(\frac{n^4(2n\gamma+3)^2}{\varepsilon^2}\right), \end{aligned} \tag{5}$$

which is polynomial for given γ and $1/\varepsilon$.

Combining the above discussion and the time complexity, we summarize our main result as the following statement.

Theorem 4. *There exists an FPTAS with the running time* $O(n^4(2ns+3)^2/\varepsilon^2)$ *for problem P*1 *or P*2.

4. Parallel-Machine Scheduling

In this paper, we will develop pseudo-polynomial-time algorithms based on dynamic programming to solve problems *P*3 and *P*4 with $m = 2$. Then, the algorithms for two parallel machines can be extended to solve general m-parallel-machine problems.

*4.1. The Nonresumable Case P*3. For a feasible partial schedule $\{J_1, \ldots, J_j\}$ where these jobs are divided into two sets, S and $\{J_1, \ldots, J_j\}/S$, we first give the following notation:

$$\begin{aligned} S &= S_1 \cup S_2, \\ \sum_{j \in S_i} a_j &= u_i \le r, \quad i = 1, 2, \end{aligned} \tag{6}$$

where u_1 (u_2): the total actual processing time of jobs finished no later than r on the first machine M_1(the second machine M_2).

Jobs in $\{J_1, \ldots, J_j\}\backslash S$ finish after r. Jobs in $\{J_1, \ldots, J_j\}\backslash S$ are divided into two groups: one is assigned to M_1 with the total processing time of the jobs v_1 and the other is assigned to M_2 with the total processing time of the jobs v_2.

$F_{[l]}$: the completion time of delivery batch B_l.

Let $F(j, u_1, u_2, v_1, v_2|B_l|)$ be the expected optimal objective value of a feasible partial schedule for jobs $J_1, \ldots, J_j$, where $|B_l|$ is the number of jobs assigned into delivery batch B_l. Set $F(j, u_1, u_2, v_1, v_2|B_l|) = \infty$, if there is no feasible schedule for the problem. The following dynamic programming algorithm (Algorithm 3) is developed to solve *P*3 with $m = 2$.

Theorem 5. *Problem P*3 *with* $m = 2$ *can be solved in time complexity* $O(n \log n + r^2 n^2 \cdot A^2)$.

Proof. We first analyze the possible recursive situations. Suppose that jobs $J_1, \ldots, J_{j-1}$ have been assigned optimally, and start to assign J_j. J_j needs to be assigned into machine 1 or machine 2. In an optimal solution, J_j on each machine will be either the last job processed no later than t or last in the whole sequence. If J_j is assigned no later than t, then it will be completed at time u_1 or u_2 on the machine. On the other hand, if J_j is assigned last in the whole sequence, then its completion time on the machine is $\sum_{\alpha=0}^{s} p_\alpha(r + \alpha + v_1)$ or $\sum_{\alpha=0}^{s} p_\alpha(r + \alpha + v_2)$. Delivery batch B_l starts at either the completion time of J_j or the return time of the transporter which has finished the previous batch to the machine. Note that if J_i is assigned into a new batch, then the delivery cost contributes D and the objective is $F_{[l]} + D$. If J_i is assigned into the current batch, the objective contribution is $F_{[l]}$. The SPT rule needs $O(n \log n)$ time to sequence the jobs in Step 1. By the definition of the recursive relations, we have $j, l \le n$, $u_1, u_2 \le r$ and $v_1, v_2 \le A$. Hence, the overall complexity is $O(n \log n + r^2 n^2\ A^2)$.

4.2. Modified Algorithm DP3. Similar to Algorithm 3 (DP3), denote u_i as the total actual processing time of jobs finished no later than r on the machine M_i, such that $\sum_{j \in S_i} a_j = u_i \le r$ for $i = 1, \ldots, m$, Denote v_i as the total actual processing time of jobs finished after r on the machine M_i, for $i = 1, \ldots, m$. Set $F(j, u_1, \ldots, u_m, v_1, \ldots, v_m, |B_l|)$ as the expected optimal objective value of a feasible partial schedule for $\{J_1, \ldots, J_j\}$, where $|B_l|$ is the number of jobs assigned into delivery batch B_l. Algorithm 3 (algorithm DP3) can be extended to the m-parallel-machine scheduling problem.

Step 1. Re-index the jobs in the SPT rule.
Step 2. Initial conditions: $F(0, 0, 0, 0, 0, 0) = 0$.
Step 3. Recursive equations:
For $j = 1, \ldots, n$; $u_1 = 0, 1, \ldots, r$; $u_2 = 0, 1, \ldots, \min\{A_j - u_1, r\}$; $v_1 = 0, 1, \ldots, A_j - u_1 - u_2$; $v_2 = A_j - u_1 - u_2 - v_1$;
For $l = 1$ and $j \le C$, set $F_{[1]} = C_j + (T/2)$ and $F(j, u_1, u_2, v_1, v_2, |B_1|) = j(C_j + (T/2)) + D$.
For $l = 2$ to n do
/* Schedule job J_j before r on machine M_1
If $|B_{l-1}| = C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), u_1\}, + (T/2)$ and $F(j, u_1, u_2, v_1, v_2, |B_l|) \longleftarrow F(j-1, u_1 - a_j, u_2, v_1, v_2, |B_{l-1}|) + F_{[l]} + D$,
If $|B_{l-1}| < C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), u_1\} + (T/2)$ and $F(j, u_1, u_2, v_1, v_2, |B_l|) \longleftarrow \min\{F(j-1, u_1 - a_j, u_2, v_1, v_2, |B_{l-1}|) + F_{[l]} + D, F(j-1, u_1 - a_j, u_2, v_1, v_2, |B_l|) + F_{[l]}\}$
Endif
/* Schedule job J_j before r on machine M_2
If $|B_{l-1}| = C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), u_2\} + (T/2)$ and $F(j, u_1, u_2, v_1, v_2, |B_l|) \longleftarrow F(j-1, u_1, u_2 - a_j, v_1, v_2, |B_{l-1}|) + F_{[l]} + D$,
If $|B_{l-1}| < C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), u_2\} + (T/2)$ and $F(j, u_1, u_2, v_1, v_2, |B_l|) \longleftarrow \min\{j-1, u_1, u_2 - a_j, v_1, v_2, |B_{l-1}| + F_{[l]} + D, F(j-1, u_1, u_2 - a_j, v_1, v_2, |B_l|) + F_{[l]}\}$
Endif
/* Schedule job J_j to be finished after r on machine M_1
If $|B_{l-1}| = C$ and $|B_l| = C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), \sum_{\alpha=0}^{s} p_\alpha (r + \alpha + v_1)\} + (T/2)$ and $F(j, u_1, u_2, v_1, v_2, |B_l|) \longleftarrow F(j-1, u_1, u_2, v_1 - a_j, v_2, |B_{l-1}|) + F_{[l]} + D$,
If $|B_{l-1}| < C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), u_1\} + (T/2)$and $F(j, u_1, u_2, v_1, v_2, |B_l|) \longleftarrow \min\{F(j-1, u_1, u_2, v_1 - a_j, v_2, |B_{l-1}|) + F_{[l]} + D, F(j-1, u_1, u_2, v_1 - a_j, v_2, |B_l|) + F_{[l]}\}$
Endif
/* Schedule job J_j to be finished after r on machine M_2
If $|B_{l-1}| = C$ and $|B_l| = C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), \sum_{\alpha=0}^{s} p_\alpha (r + \alpha + v_2)\} + (T/2)$ and $F(j, u_1, u_2, v_1, v_2, |B_l|) \longleftarrow F(j-1, u_1, u_2, v_1, v_2 - a_j, |B_{l-1}|) + F_{[l]} + D$,
If $|B_{l-1}| < C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), \sum_{\alpha=0}^{s} p_\alpha (r + \alpha + v_2)\} + (T/2)$ and $F(j, u_1, u_2, v_1, v_2, |B_l|) \longleftarrow \min\{F(j-1, u_1, u_2, v_1, v_2 - a_j, |B_{l-1}|) + F_{[l]} + D, F(j-1, u_1, u_2, v_1, v_2 - a_j, |B_l|) + F_{[l]}\}$
Endfor
Endfor
Step 4. Optimal solution:
$\min\{F(n, u_1, u_2, v_1, v_2, |B_l|) : \text{for all possible } u_1, u_2, v_1, v_2, l, \lceil n/C \rceil \le l \le n\}$.

Algorithm 3: Algorithm DP3.

For $l = 1$ and $j \le C$, set $F(j, u_1, ..., u_m, v_1, ..., v_m, |B_1|) = j(C_j + (T/2)) + D$.
For $l = 2$ to n do,
/*Schedule job J_j to be finished after r on machine M_i, for $i = 1, \ldots, k$
If $|B_{l-1}| = C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), u_1\} + (T/2)$ and $F(j, u_1, \ldots, u_m, v_1, \ldots, v_m, |B_l|) \longleftarrow F(j-1, u_1, \ldots, u_i - a_j, \ldots, u_m, v_1, \ldots, v_m, |B_{l-1}|) + F_{[l]} + D$
If $|B_{l-1}| < C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), u_i\} + (T/2)$ and $F(j, u_1, \ldots, u_m, v_1, \ldots, v_m, |B_l|) \longleftarrow \min\{F(j-1, u_1, \ldots u_i - a_j, \ldots, u_m, v_1, \ldots, v_m | B_{l-1}|) + F_{[l]} + D, F(j-1, u_1, \ldots, u_i - a_j, \ldots, u_m, v_1, \ldots, v_m | B_l|) + F_{[l]}\}$
Endif
/*Schedule job J_j to be finished after r on machine M_i, for $i = 1, \ldots, k$
If $|B_{l-1}| = C$ and $|B_l| = C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), \sum_{\alpha=0}^{s} p_\alpha (r + \alpha + v_i)\} + (T/2)$ and $F(j, u_1, \ldots, u_m, v_1, \ldots, v_m, |B_l|) \longleftarrow F(j-1, u_1, \ldots, u_m, v_1, \ldots, v_i - a_j, \ldots, v_m, |B_{l-1}|) + F_{[l]} + D$
If $|B_{l-1}| < C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), \sum_{\alpha=0}^{s} p_\alpha (r + \alpha + v_i)\} + (T/2)$ and $F(j, u_1, \ldots, u_m, v_1, \ldots, v_m, |B_l|) \longleftarrow \min\{F(j-1, u_1, \ldots, u_m, v_1, \ldots, v_i - a_j, \ldots, v_m, |B_{l-1}|) + F_{[l]} + D, F + (j-1, u_1, \ldots, u_m, v_1, \ldots, v_i - a_j, \ldots, v_m, |B_l||) + F_{[l]}\}$
Endfor
Optimal solution:
$\min\{F(n, u_1, \ldots, u_m, v_1, \ldots, v_m, |B_l|)$:
for all possible $u_1, \ldots, u_m, v_1, \ldots, v_m, l, \lceil n/C \rceil \le l \le n\}$.

Time complexity: the SPT rule needs $O(n \log n)$ time to sequence the jobs. Because $j, l \le n$, $u_1, \ldots, u_m \le r$, and $v_1, \ldots, v_m \le A$, the overall complexity of modified algorithm DP3 to solve m-parallel-machine problem is $O(n \log n + r^m n^2 \cdot A^m)$.

4.3. The Resumable Case P4. Similar to the nonresumable case $P3$, a pseudo-polynomial-time algorithm based on dynamic programming is developed for $P4$ with $m = 2$.

For a feasible partial schedule $\{J_1, \ldots, J_j\}$ where these jobs are divided into two sets, S and $\{J_1, \ldots, J_j\}/S$, we first give the following notation: $S = S_1 \cup S_2$ and $\sum_{j \in S_i} a_j = u_i \le r$ for $i = 1, 2$.

Step 1. Re-index the jobs in the SPT rule.
Step 2. Initial conditions: $F(0, 0, 0, 0, 0, 0, 0, 0) = 0$.
Step 3. Recursive equations:
For $j = 1, \ldots, n$; $u_1 = 0, 1, \ldots, r$; $u_2 = 0, 1, \ldots, \min\{A_j - u_1, r\}$; $v_1 = 0, 1, \ldots, A_j - u_1 - u_2$; $v_2 = A_j - u_1 - u_2 - v_1 =$; $q_1 = \max\{u_1, r - a_{\max} + 1\}, \ldots, r$; $q_2 = \max\{u_2, r - a_{\max} + 1\}, \ldots, r$.
For $l = 1$ and $j \le C$, then $F(j, u_1, u_2, q_1, q_2, v_1, v_2, |B_1|) = j(C_j + (T/2)) + D$.
For $l = 2$ to n do,
/* schedule job J_j before r on machine M_2
If $|B_{l-1}| = C$ and $|B_l| = C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), u_1\} + (T/2)$ and $F(j, u_1, u_2, q_1, q_2, v_1, v_2, |B_l|) \longleftarrow F(j-1, u_1 - a_j, u_2, q_1, q_2, v_1, v_2, |B_{l-1}|) + F_{[l]} + D$,
If $|B_{l-1}| < C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), u_1\} + (T/2)$ and $F(j, u_1, u_2, q_1, q_2, v_1, v_2, |B_l|) \longleftarrow \min\{F(j-1, u_1 - a_j, u_2, q_1, q_2, v_1, v_2, |B_{l-1}|) + F_{[l]} + D, F(j-1, u_1 - a_j, u_2, q_1, q_2, v_1, v_2, |B_l|) + F_{[l]}\}$
/* schedule job J_j before r on machine M_2
If $|B_{l-1}| = C$ and $|B_l| = C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), u_2\} + (T/2)$ and $F(j, u_1, u_2, q_1, q_2, v_1, v_2, (B_l)) \longleftarrow F(j-1, u_1, u_2 - a_j, q_1, q_2, v_1, v_2, |B_{l-1}|) + F_{[l]} + D$,
If $|B_{l-1}| < C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), u_2\} + (T/2)$ and $F(j, u_1, u_2, q_1, q_2, v_1, v_2, |B_l|) \longleftarrow \min\{F(j-1, u_1, u_2 - a_j, q_1, q_2, v_1, v_2, |B_{l-1}|) + F_{[l]} + D, F(j-1, u_1, u_2 - a_j, q_1, q_2, v_1, v_2, |B_l|) + F_{[l]}\}$
Endif
/* schedule job J_j to be finished after r on machine M_1
If $|B_{l-1}| = C$ and $|B_l| = C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), \sum_{\alpha=0}^{s} p_\alpha (q_1 + \alpha + v_1)\} + (T/2)$ and $F(j, u_1, u_2, q_1, q_2, v_1, v_2, |B_l|) \longleftarrow F(j-1, u_1, u_2, q_1, q_2, v_1 - a_j, v_2, |B_{l-1}|) + F_{[l]} + D$,
If $|B_{l-1}| < C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), \sum_{\alpha=0}^{s} p_\alpha (q_1 + \alpha + v_1)\} + (T/2)$ and $F(j, u_1, u_2, q_1, q_2, v_1, v_2, |B_l|) \longleftarrow \min\{F(j-1, u_1, u_2, q_1, q_2, v_1 - a_j, v_2, |B_{l-1}|) + F_{[l]} + D, F(j-1, u_1, u_2, q_1, q_2, v_1 - a_j, v_2, |B_l|) + F_{[l]}\}$
Endif
/* schedule job J_j to be finished after r on machine M_2
If $|B_{l-1}| = C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), \sum_{\alpha=0}^{s} p_\alpha (q_2 + \alpha + v_2)\} + (T/2)$ and $F(j, u_1, u_2, q_1, q_2, v_1, v_2, |B_l|) \longleftarrow F(j-1, u_1, u_2, q_1, q_2, v_1, v_2 - a_j, |B_{l-1}|) + F_{[l]} + D$,
If $|B_{l-1}| < C$ and $|B_l| < C$, then set $F_{[l]} = \max\{F_{[l-1]} + (T/2), \sum_{\alpha=0}^{s} p_\alpha (q_2 + \alpha + v_2)\} + (T/2)$ and $F(j, u_1, u_2, q_1, q_2, v_1, v_2, |B_l|) \longleftarrow \min\{F(j-1, u_1, u_2, q_1, q_2, v_1, v_2 - a_j, |B_{l-1}|) + F_{[l]} + D, F(j-1, u_1, u_2, q_1, q_2, v_1, v_2 - a_j, |B_l|) + F_{[l]}\}$
Endfor
Endfor
Step 4. Optimal solution:
$\min\{F(n, u_1, u_2, q_1, q_2, v_1, v_2, |B_l|)$: for all possible $u_1, u_2, q_1, q_2, v_1, v_2, l, \lceil n/C \rceil \le l \le n\}$.

Algorithm 4: Algorithm DP4.

Jobs in $\{J_1, \ldots, J_j\} \backslash S$ are finished processing after r. And jobs in $\{J_1, \ldots, J_j\} \backslash S$ are divided into two groups: one is assigned to M_1 with the total processing time v_1, and starts at time q_1, while the other is assigned to M_2 with the total processing time v_2 and starts at time q_2. If there is no feasible solution for $P4$, then $F(j, u_1, u_2, q_1, q_2, v_1, v_2, |B_l|) = \infty$. Here, q_1 and q_2 represent the state link between the set of jobs finished no later than r and the set of jobs finished after r.

Time complexity: the SPT rule needs $O(n \log n)$ time to sequence the jobs in Step 1. By the definition of the recursive relation, we have $j, l \le n$, $u_1, u_2, q_1, q_2 \le r$ and $v_1, v_2 \le A$. Hence, the overall complexity is $O(n \log n + r^4 n^2 \cdot A^2)$.

Theorem 6. *Problem P4 with $m = 2$ can be solved in time complexity $O(n \log n + r^4 n^2 \cdot A^2)$.*

Similar to modified algorithm DP3, define $F(n, u_1, \ldots, u_m, q_1, \ldots, q_m, v_1, \ldots, v_m, |B_l|)$ as the objective value for $P4$, and we can extend Algorithm 4 to solve general m-parallel-machine problem with resumable case. The overall time complexity to solve $P4$ is $O(n \log n + r^{2m} n^2 \cdot A^m)$.

Theorem 7. *$P3$ and $P4$ are NP-hard in the ordinary sense.*

4.4. Fully Polynomial-Time Approximation Schemes for $P3$ and $P4$. In this section, the dynamic programming algorithms for $P3$ or $P4$ will be translated into fully polynomial-time approximation schemes. The main framework is to iteratively thin out the state pace of the dynamic programming by using the trimming technique. This method is to collapse solutions that are close to each other and decrease the size of the state space down to polynomial.

4.4.1. FPTAS for $P3$. In the dynamic programming DP 3, we will store necessary information for some schedules for the first j jobs: each schedule is indexed by a six-dimensional vector $[u_1, u_2, v_1, v_2, l, F]$. Let the trimming parameter be $\delta = 1 + (\varepsilon/2n)$ for any given $\varepsilon > 0$. The trimming of the state spaces is formulated as the notion of δ-domination: a state $s' = [u_1', u_2', v_2', v_2', l.F']$ is δ-dominated by another state $s = [u_1, u_2, v_1, v_2, l, F]$, if and only if

$$\begin{aligned}
\frac{T}{\delta} &\le T' \le T\delta,\\
\frac{u_1}{\delta} &\le u_1' \le u_1\delta,\\
\frac{u_2}{\delta} &\le u_2' \le u_2\delta,\\
\frac{v_1}{\delta} &\le v_1' \le v_1\delta,\\
\frac{v_2}{\delta} &\le v_2' \le v_2\delta,\\
\frac{F}{\delta} &\le F' \le F\delta.
\end{aligned} \tag{7}$$

If a state s' contained in the state space is δ-dominated by another state s, then the dominated state s' can be removed. The trimming state space forms if the removal procedure eventually stops. When a new state space in the trimmed dynamic programming is computed, the trimmed state space can start instead of the original state space in the original dynamic programming. Next, we will present the time complexity of the trimmed state space.

Note that there are at most r possible values for each of u_1 and u_2: A possible values for each of v_1 and v_2 and at most $n^2r^2A^2$ possible values for F. For any vector $(u_1, u_2, v_1, v_2, l, F)$ of integers, the trimmed state space contains at most one state $[u_1, u_2, v_1, v_2, l, F]$ with $\delta^{f_1} \le u_1 \le \delta^{f_1+1}$, $\delta^{f_2} \le u_2 \le \delta^{f_2+1}$, $\delta^{g_1} \le v_1 \le \delta^{g_1+1}$, and $\delta^{g_2} \le v_2 \le \delta^{g_2+1}$. Since $f_1 f_2 \le \lceil \log_\delta r \rceil$ and $g_1 g_2 \le \lceil \log_\delta A \rceil$, up to a constant factor the time complexity of a state space is bounded from above by

$$\begin{aligned}
O\left(n^2 (\log_\delta r)^2 (\log_\delta A)^2 + n\log n\right) &\le O\left(n^2 \left(\frac{\log_2 r}{\log_2 \delta}\right)^2 \left(\frac{\log_2 A}{\log_2 \delta}\right)^2\right),\\
&\le O\left(n^2 \frac{(\log_2 r)^2 (\log_2 A)^2}{(\log_2 \delta)^4}\right) \le O\left(n^2 \frac{(\log_2 r)^2 (\log_2 A)^2}{(\delta - 1)^4}\right) \le O\left(\frac{n^6 (\log_2 r)^2 (\log_2 A)^2}{\varepsilon^4}\right).
\end{aligned} \tag{8}$$

Here, the inequality $\log_2(1 + x) \ge x$ for $0 \le x \le 1$ and $\delta = 1 + (\varepsilon/2n)$. Hence, the time complexity of the space is polynomial bounded in the input size and in $1/\varepsilon$.

Theorem 8. *For P3, the best feasible solution for the trimmed problem has an objective value that is at most a factor* $1 + \varepsilon$ *above the objective of the best feasible solution for the original problem.*

Proof. Note that the final trimmed state $s' = [u_1', u_2', v_2', v_2', l.F']$ yields the optimal objective value. The trimmed state space contains a state $s = [u_1, u_2, v_1, v_2, l, F]$ that is δ^n-dominated by s'. Hence, the total cost F is at most a factor $\delta^n = (1 + (\varepsilon/2n))^n \le 1 + \varepsilon$ above the optimal cost where the inequality $(1 + (x/n))^n \le 1 + 2x$ is used for real number x with $0 \le x \le 1$ and for integers $n \ge 1$.

4.4.2. FPTAS for P4. Similar to the above discussion for $P3$, each schedule is indexed by an eight-dimensional vector $[u_1, u_2, v_1, v_2, l, F]$. A state $s' = [u_1', u_2', q_1', q_2'v_2', v_2', l.F']$ is δ-dominated by another state $s = [j, u_1, u_2, q_1, q_2, v_1, v_2, l]$ if and only if

$$\begin{aligned}
\frac{T}{\delta} &\le T' \le T\delta,\\
\frac{u_1}{\delta} &\le u_1' \le u_1\delta,\\
\frac{u_2}{\delta} &\le u_2' \le u_2\delta,\\
\frac{q_1}{\delta} &\le q_1' \le q_1\delta,\\
\frac{q_2}{\delta} &\le q_2' \le q_2\delta,\\
\frac{v_1}{\delta} &\le v_1' \le v_1\delta,\\
\frac{v_2}{\delta} &\le v_2' \le v_2\delta.
\end{aligned} \tag{9}$$

The worst-case discussion is similar to Theorem 5. The time complexity of a trimmed state space is bounded from above by

$$
\begin{aligned}
&O\left(n^2(\log_\delta r)^4(\log_\delta A)^2 + n\log n\right)\\
&\le O\left(n^2\left(\frac{\log_2 r}{\log_2 \delta}\right)^4\left(\frac{\log_2 A}{\log_2 \delta}\right)^2\right),\\
&\le O\left(n^2\frac{(\log_2 r)^4(\log_2 A)^2}{(\log_2 \delta)^6}\right) \le O\left(n^2\frac{(\log_2 r)^2(\log_2 A)^2}{(\delta-1)^6}\right)\\
&\le O\left(\frac{n^6(\log_2 r)^2(\log_2 A)^2}{\varepsilon^6}\right).
\end{aligned}
\tag{10}
$$

Hence, the time complexity of the trimmed state spaces in the dynamic programming DP 3 or DP 4 can be done within a time polynomial complexity bounded in $1/\varepsilon$. The worst-case ratio guarantees $1+\varepsilon$. We have the following theorem.

Theorem 9. *Problem P*3 *and P*4 *have fully polynomial-time approximation schemes, respectively.*

5. Conclusions

In this paper, we study single-machine and two-parallel-machine scheduling problems with batch deliveries and potential disruption motivated by the iron and steel industry. The objective is to find a coordinated schedule to minimize the total expected flow time plus the total delivery cost. For the four corresponding scheduling problems, pseudo-polynomial-time algorithms and fully polynomial-time approximation schemes are presented in this paper, respectively. Furthermore, the existence of the FPTAS means that the strongest possible polynomial-time approximation result is obtained for a NP-hard problem. In addition, several issues are worthy of future investigations. A future interesting issue is to develop effective heuristics or reinforcement learning algorithm to solve the general problem with stochastic processing times. It would also be interesting to study the problem by taking into account other objective functions.

Data Availability

No data were used to support the study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Hua Gong and Puyu Yuan contributed equally to this study.

Acknowledgments

This research was partly supported by the Project of Liaoning BaiQianWan Talents Program (grant no. LR201945-22), the Science and Technology Project of Educational Department of Liaoning Province (grant no. LG201912), the Major International Joint Research Project of the National Natural Science Foundation of China (71520107004), and the 111 Project (B16009).

References

[1] Z.-L. Chen, "Integrated production and outbound distribution scheduling: review and extensions," *Operations Research*, vol. 58, no. 1, pp. 130–148, 2010.

[2] D. Y. Wang, O. Grunder, and A. E. Moudni, "Integrated scheduling of production and distribution operations: a review," *International Journal of Industrial and Systems Engineering*, vol. 19, no. 1, pp. 94–122, 2015.

[3] F. Li, Z.-L. Chen, and L. Tang, "Integrated production, inventory and delivery problems: complexity and algorithms," *Informs Journal on Computing*, vol. 29, no. 2, pp. 232–250, 2017.

[4] L. Tang, F. Li, and J. Liu, "Integrated scheduling of loading and transportation with tractors and semitrailers separated," *Naval Research Logistics (NRL)*, vol. 62, no. 5, pp. 416–433, 2015.

[5] L. Tang, H. Gong, J. Liu, and F. Li, "Bicriteria scheduling on a single batching machine with job transportation and deterioration considerations," *Naval Research Logistics (NRL)*, vol. 61, no. 4, pp. 269–285, 2014.

[6] H. Gong, L. Tang, and J. Y. T. Leung, "Parallel machine scheduling with batch deliveries to minimize total flow time and delivery cost," *Naval Research Logistics (NRL)*, vol. 63, no. 6, pp. 492–502, 2016.

[7] N. G. Hall and C. N. Potts, "The coordination of scheduling and batch deliveries," *Annals of Operations Research*, vol. 135, no. 1, pp. 41–64, 2005.

[8] G. Wang and T. C. E. Cheng, "Parallel machine scheduling with batch delivery costs," *International Journal of Production Economics*, vol. 68, no. 2, pp. 177–183, 2000.

[9] Z.-L. Chen and G. L. Vairaktarakis, "Integrated scheduling of production and distribution operations," *Management Science*, vol. 51, no. 4, pp. 614–628, 2005.

[10] L. Wan and A. Zhang, "Coordinated scheduling on parallel machines with batch delivery," *International Journal of Production Economics*, vol. 150, no. 4, pp. 199–203, 2014.

[11] C.-Y. Lee and G. Yu, "Single machine scheduling under potential disruption," *Operations Research Letters*, vol. 35, no. 4, pp. 541–548, 2007.

[12] C. Y. Lee and G. Yu, "Parallel-machine scheduling under potential disruption," *Optimization Letters*, vol. 2, no. 1, pp. 27–37, 2008.

[13] Y. Yin, Y. Wang, T. C. E. Cheng, W. Liu, and J. Li, "Parallel-machine scheduling of deteriorating jobs with potential machine disruptions," *Omega*, vol. 69, no. 6, pp. 17–28, 2017.

[14] B. Zheng and M. Fan, "Parallel-machine scheduling with potential disruption and positional-dependent processing times," *Journal of Industrial and Management Optimization*, vol. 13, no. 2, pp. 697–711, 2017.

[15] C.-Y. Lee, J. Y.-T. Leung, and G. Yu, "Two machine scheduling under disruptions with transportation considerations," *Journal of Scheduling*, vol. 9, no. 1, pp. 35–48, 2006.

[16] M. L. Pinedo, *Scheduling Theory, Algorithms, and Systems*, Springer, New York City, NY, USA, 2010.

[17] G. J. Woeginger, "When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (fptas)?" *INFORMS Journal on Computing*, vol. 12, no. 1, pp. 57–74, 2000.