Omega ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

Parallel-machine scheduling of deteriorating jobs with potential machine disruptions ☆

Yunqiang Yin [a,*], Yan Wang [a], T.C.E. Cheng [b], Wenqi Liu [a], Jinhai Li [a]

[a] Faculty of Science, Kunming University of Science and Technology, Kunming, China
[b] Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

ARTICLE INFO

Article history:
Received 10 August 2015
Accepted 27 July 2016

Keywords:
Scheduling
Deteriorating jobs
Disruptive environment
Fully polynomial-time approximation scheme

ABSTRACT

We consider parallel-machine scheduling of deteriorating jobs in a disruptive environment in which some of the machines will become unavailable due to potential disruptions. This means that a disruption to some of the machines may occur at a particular time, which will last for a period of time with a certain probability. If a job is disrupted during processing by a disrupted machine and it does not need (needs) to re-start after the machine becomes available again, it is called the resumable (non-resumable) case. By deteriorating jobs, we mean that the actual processing time of a job grows when it is scheduled for processing later because the machine efficiency deteriorates over time due to machine usage and aging. However, a repaired machine will return to its original state of efficiency. We consider two cases, namely performing maintenance immediately on the disrupted machine when a disruption occurs and not performing machine maintenance. In each case, the objective is to determine the optimal schedule to minimize the expected total completion time of the jobs in both non-resumable and resumable cases. We determine the computational complexity status of various cases of the problem, and provide pseudo-polynomial-time solution algorithms and fully polynomial-time approximation schemes for them, if viable.

1. Introduction

Contemporary production and service systems often operate in a dynamic and uncertain environment, in which unexpected events may occur from time to time. Should the expected events be *disruptions*, they may cause some resources (machines or facilities) to be unavailable for a certain period of time, which will directly affect the utilization of the resources and ultimately customer service. Examples of disruptive events occurring during production abound, e.g., machine breakdowns, power failures, and shortages of raw materials, personnel, tools, etc. Research on scheduling that takes disruptions into account is commonly known as scheduling with availability constraints, which has been extensively investigated in the literature. Lee et al. [13], Sanlaville and Schmidt [23], Schmidt [25], and Ma et al. [19] survey and summarize the major results and practices in this area.

Machine scheduling with availability constraints can be categorized into two major classes. One class is where the machine unavailability is deterministic due to some internal factors such as preventive maintenance. In this case, both the disruption starting time and duration are either fixed in advance [4,8–11,20,31,34,33] or are decision variables in the scheduling model [5,18,21,22,28–30,32]. The other class is where the machine unavailability is stochastic [1–3], which is caused by machine breakdowns or other internal and external factors. Lee and Yu [15] consider single-machine scheduling with potential disruptions due to external factors, e.g., bad weather (typhoons and snowstorms), labour strikes, power shortages, etc. In such a case, the disruption starting time is roughly known (should it happen); however, the disruption duration is unknown until the damage is made. They provide pseudo-polynomial-time algorithms to solve the problems of minimizing the expected total weighted completion time and the expected maximum tardiness. Subsequently, Lee and Yu [16] extend the results to the parallel-machine case to minimize the expected total weighted completion time.

☆This manuscript was processed by Associate Editor Strusevich.

* Corresponding author.
E-mail address: yinyunqiang@126.com (Y. Yin).

http://dx.doi.org/10.1016/j.omega.2016.07.006
0305-0483/

We pursue the stream of research initiated by Lee and Yu [15,16]. We consider scheduling of jobs on m identical parallel machines that are subject to potential disruptions in a deteriorating production environment, which means that the job processing times will deteriorate over time. Some of the machines may become unavailable for a period of time over the scheduling period due to potential disruptions arising from worker shortage, power shortage, etc. In such a case, we often know the disruption starting time (should it happen) in advance, yet the duration is unknown until it happens. That is, there is a possibility that a disruption will happen at a particular time and the disruption will last for a certain duration with a certain probability. So the machine unavailability will only be revealed at the time when the disruption occurs. Thus we assume that once a disruption occurs, we will know its duration. Specifically, we consider two cases. One is to perform maintenance immediately on each of the disrupted machines when a disruption occurs and the other is not to perform machine maintenance, where performing machine maintenance will improve the efficiency of the machine by returning it to its original state of efficiency at the expense of the cost incurred from maintenance. With known probabilities of all the unexpected events, the scheduling objective is to find an optimal schedule for the jobs to minimize the expected total completion time of the jobs. We extend the work of Lee and Yu [16] in three major ways as follows:

- We consider the scheduling problem in a deteriorating production environment, i.e., the actual processing time of a job grows when it is scheduled for processing later because the machine efficiency deteriorates over time due to machine usage and aging, which more accurately reflects real-life production.
- We assume that machine unavailability will only occur on some of the machines, which is the case where the factory has backup power to keep some of the machines working when the disruption occurs due to power shortage, whereas Lee and Yu [16] assume that machine unavailability will happen on all the machines.
- We include the case where the disruption may not happen (i.e., $\zeta_\gamma = 0$) in the non-resumable case, which Lee and Yu [16] do not consider.

The purpose of this paper is twofold. One is to study a more realistic and complex scheduling model that takes both potential machine disruptions and job deterioration into consideration. The other is to ascertain the computational complexity status and provide solution procedures, if viable, for the problems under consideration.

The rest of the paper is organized as follows: In Section 2 we introduce the notation and formally formulate our problems. In Section 3 we derive some structural properties of the optimal solutions that are useful for tackling our problems. In Sections 4 and 5 we analyze the computational complexity status and provide solution procedures, if viable, for the problems in the non-resumable and resumable cases, respectively. In the last section we conclude the paper and suggest topics for future research.

2. Problem formulation

We formally describe the general problem under consideration as follows: There are n independent jobs in the job set $N = \{J_1, J_2, \ldots, J_n\}$ to be processed on m identical parallel machines $M_1, M_2, \ldots, M_m$ over a scheduling period T. All the n jobs are available for processing at time zero. Each job needs to be processed on one machine only and each machine is capable of processing any job but at most one job at a time. The machines will experience deterioration in efficiency due to usage and aging [26]. As a result of deterioration in machine efficiency over time, the actual processing time of a job will become longer if it is scheduled for processing later. Specifically, if job J_j is processed on machine M_i and starts processing at time t, we define its actual processing time as $p_{jt} = p_j(1+at)$, where p_j is the normal processing time of job J_j and a ($a > 0$) is the deteriorating rate common to all the jobs. Some of the m machines may become unavailable due to potential machine disruptions, each of which will last for a period of time with a certain probability. Without loss of generality, we assume that a machine disruption will happen at time r, which makes the first ι machines $M_1, \ldots, M_\iota$, $1 \le \iota \le m$, unavailable and the duration will take γ ($\gamma = 0, 1, \ldots, s$) time units with a probability ζ_γ, which is the same for all the disrupted machines once the anticipated disruption occurs. Here, ζ_0 is the probability that the disruption will not happen and s is the maximum possible duration.

In order to reduce the effect of machine deterioration, an option strategy is to perform machine maintenance, which will improve the efficiency of the machine by making it return to its original state of efficiency. It follows that the actual processing time of job J_j will be the same as its normal processing time if it is the first job that starts processing on a repaired machine after maintenance. However, we focus on the case where the effect of machine deterioration on the objective value during the scheduling period is smaller compared with the cost incurred from the maintenance duration and the maintenance cost, which is reasonable in most production and service systems. As a consequence, we consider the following two cases:

Case 1: Perform maintenance immediately on each of the disrupted machines with a fixed duration, denoted by κ, when a disruption occurs by several maintenance workers (or teams). Thus, the disrupted machines will become unavailable during the time interval $[r, r+\max\{\gamma, \kappa\}]$ under the scenario γ ($\gamma \ge 1$).

Case 2: Do not perform machine maintenance.

The reason for considering Case 1 is twofold. One is that it may reduce the effect on the objective value caused by machine disruptions because performing maintenance can improve the efficiency of the machines. The other is that it may reduce the effect on the objective value caused by performing maintenance because disrupted machines have unavailable time intervals due to machine disruptions, which can reduce the effect from maintenance duration. In each case, we consider both the *resumable* and *non-resumable* cases. If a job is disrupted during processing by a disrupted machine and it does not need (needs) to re-start after the machine becomes available again, it is called the resumable (non-resumable) case (see [12]). Assume that the last job started before but not completed at r is job J_j. In the non-resumable case, J_j needs to re-start at times $r+\max\{\gamma, \kappa\}$ and $r+\gamma$ for Case 1 and Case 2, respectively, under scenario $\gamma \ge 1$, while in the resumable case, processing of the remaining part of J_j will continue at times $r+\max\{\gamma, \kappa\}$ and $r+\gamma$ for Case 1 and Case 2, respectively, without any penalty.

We assume throughout the paper that p_j, r, s, and κ are known positive integers such that $\kappa \le s$ and a is chosen such that ap_j is a positive integer for all $j = 1, \ldots, n$. The objective is to determine an optimal schedule to minimize the expected total completion time of the jobs, i.e., $E(\sum_{j=1}^{n} C_j)$, where C_j denotes the completion time of job J_j in a given sequence.

Please cite this article as: Yin Y, et al. Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.07.006

Using the three-field notation $\alpha|\beta|\gamma$ introduced by Graham et al. [6] for describing scheduling problems, we denote our problem by $Px,\iota|r/nr-a,M/NM,DE,PDR|E(\sum_{j=1}^{n}C_j)$, where $\alpha=Px,\iota$ denotes that there are m identical parallel machines, in which x is empty when m is considered to be part of the input and $x=m$ when m is fixed, and the first ι machines will become unavailable if an anticipated disruption happens; $r-a$ and $nr-a$ denote the resumable and non-resumable cases, respectively; M and NM denote performing maintenance immediately on the disrupted machines when an anticipated disruption occurs and not performing machine maintenance, respectively; DE denotes job deterioration; and PDR indicates potential disruptions.

3. Preliminary analysis

In this section we derive some structural properties of the optimal schedules for the problems under study, which we will use for the design of solution algorithms in the following sections. Let us first recall some results on the corresponding single-machine problem without machine disruptions.

Lemma 3.1 ([27]). *For the problem $1|DE|\theta$, where θ is any regular scheduling measure to be minimized, the completion time of the jth job in the sequence $(J_{[1]},\ldots,J_{[j-1]},J_{[j]},J_{[j+1]},\ldots,J_{[n]})$ is equal to $C_{[j]}=(s_0+\frac{1}{a})\prod_{k=1}^{j}\left(1+ap_{[k]}\right)-\frac{1}{a}$ if the first job starts processing at time s_0.*

Lemma 3.2 ([27]). *For the problem $1|DE|\sum C_j$, there exists an optimal schedule in which the jobs are scheduled in the shortest processing time (SPT) order.*

As a consequence of Lemma 3.2, the following result holds:

Lemma 3.3. *For the problem $Px,\iota|r/nr-a,M/NM,DE,PDR|E\left(\sum_{j=1}^{n}C_j\right)$, there exists an optimal schedule in which*

(1) *the jobs finished no later than r on each disrupted machine M_i, $i=1,\ldots,\iota$, and the jobs processed on each non-disrupted machine M_i, $i=\iota+1,\ldots,m$, are sequenced in the SPT order, respectively; and*
(2) *the jobs started no earlier than r on each disrupted machine M_i, $i=1,\ldots,\iota$, are sequenced in the SPT order.*

Proof. It follows from Lemma 3.2 that the arguments hold for all possible scenarios of γ and all the machines with the objective of minimizing the total completion time. Thus, the lemma holds for the objective of minimizing the expected total completion time $E\left(\sum_{j=1}^{n}C_j\right)$. □

Thus, in the rest of this paper, we assume, without loss of generality, that the jobs are re-indexed in the SPT order such that $p_1\le\cdots\le p_n$. For notational convenience, we let $p_{\max}=\max_{j=1,\ldots,n}p_j$, $TP=\frac{1}{a}\prod_{j=1}^{n}(1+ap_j)-\frac{1}{a}$, and $\mathcal{TP}=(r+s+\frac{1}{a})\prod_{j=1}^{n}(1+ap_j)-\frac{1}{a}$. We assume that $r<TP\le T$ for the case of performing machine maintenance and $\mathcal{TP}\le T$ for the case of not performing machine maintenance. Otherwise, the potential machine disruptions have no effect on the solution.

4. The non-resumable case

Since even for the case without deteriorating effect with $\iota=m$ and $\zeta_s=1$, the problem with an arbitrary m, denoted as $P,h_{i1}|nr-a|\sum_{j=1}^{n}C_j$, is NP-hard in the strong sense [17], our problem when both m and ι are arbitrary, denoted as $P,\iota|nr-a,M/NM,DE,PDR|E\left(\sum_{j=1}^{n}C_j\right)$, is NP-hard in the strong sense, too. Note also that Lee and Liman [14] show that the problem without deteriorating effect on a single machine with $\zeta_s=1$ is NP-hard, which implies that our problem is also NP-hard when m is fixed. Hence, we focus on the case with a fixed m in this section. We first design a pseudo-polynomial-time solution algorithm for each of the cases of performing and not performing machine maintenance, establishing that it is NP-hard in the ordinary sense. We then show how to convert the solution algorithm into a fully polynomial-time approximation scheme (FPTAS) for the special case with $\iota=1$. Recall that a solution algorithm A_ε for a minimization problem is a $(1+\varepsilon)$-approximation algorithm if it always delivers an approximate solution Z with $Z\le(1+\varepsilon)Z^*$ for all the instances, where Z^* is the optimal solution value. A family of approximation algorithms $\{A_\varepsilon\}$ defines an FPTAS for the considered problem, if for any $\varepsilon>0$, A_ε is a $(1+\varepsilon)$-approximation algorithm that is polynomial in n, L, and $1/\varepsilon$, where $L=\log\max\{n,r,s,ap_{\max}\}$ is the number of bits in binary encoding for the largest numerical parameter in the input.

4.1. Pseudo-polynomial-time algorithms

In this subsection we develop pseudo-polynomial-time dynamic programming algorithms to solve the problems $Pm,\iota|nr-a,M,DE,PDR|E\left(\sum_{j=1}^{n}C_j\right)$ and $Pm,\iota|nr-a,NM,DE,PDR|E\left(\sum_{j=1}^{n}C_j\right)$, respectively.

4.1.1. The problem $Pm,\iota|nr-a,M,DE,PDR|E\left(\sum_{j=1}^{n}C_j\right)$

This subsection focuses on the case of performing machine maintenance. For simplicity, we only analyze the two-machine case with $\iota=1$ in detail, and then briefly discuss how to generalize the results to the case with m machines for any given ι.

We first develop a forward dynamic programming algorithm *NRMDPDDP* for the case where $\zeta_0\ne0$. The algorithm first fixes the variable t_1, which denotes the starting time of the first job scheduled on machine M_1 that will finish after r in the final optimal schedule when disruption does not happen on machine M_1. This means that if disruption does not happen, the first job scheduled on machine M_1 that will finish after r starts at time t_1, and if the anticipated disruption happens and lasts for a time period γ ($\gamma>0$), the first job scheduled on machine M_1 that will finish after r will restart at time $r+\max\{\gamma,\kappa\}$, even though it may have been processed during $[t_1,r]$. Due to the deteriorating effect, it is easy to see that t_1 ranges from τ to r, where $\tau=\lceil\frac{r-p_{\max}}{1+ap_{\max}}\rceil$. For each given t_1, the algorithm consists of $n+1$ phases and in each phase j, $j=0,1,\ldots,n$, a state space $\mathcal{F}_{(j,t_1)}$ is generated. Any state in $\mathcal{F}_{(j,t_1)}$ is a vector (u_1,u_2,v_1,q_1,f) that encodes a feasible partial schedule for the jobs $\{J_1,\ldots,J_j\}$, where the variables u_1 ($0\le u_1\le t_1$) and u_2 denote the total actual processing time of the jobs finished no later than r on machine M_1 and the total

Please cite this article as: Yin Y, et al. Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.07.006

actual processing time of the jobs processed on machine M_2, respectively, v_1 and q_1 measure the total actual processing times of the jobs finished after r on machine M_1 for the cases where the disruption does not happen and the anticipated disruption does happen on machine M_1, respectively, and f stands for the expected total completion time of the partial schedule. The state spaces $\mathcal{F}_{(j,t_1)}$, $j=0,1,\ldots,n$, $t_1=\tau,\ldots,r$, are constructed iteratively, where the initial space $\mathcal{F}_{(0,t_1)}$ for each $t_1=\tau,\ldots,r$ contains $(0,0,0,0,0)$ as its only element. For each given t_1, in the jth phase, $j=1,\ldots,n$, we build a state by adding a single job J_j to a previous state, if it is possible for the given state. To add job J_j to a state $(u_1,u_2,v_1,q_1,f)\in\mathcal{F}_{(j-1,t_1)}$, there are three cases to consider as follows:

Case 1: Schedule job J_j before t_1 on machine M_1. This is possible only when $u_1+p_j(1+au_1)\le t_1$. In this case, the contribution of job J_j to the objective function is $u_1+p_j(1+au_1)$. Hence, we include the new state $(u_1+p_j(1+au_1),u_2,v_1,q_1,f+u_1+p_j(1+au_1))$ in $\mathcal{F}_{(j,t_1)}$ if $u_1+p_j(1+au_1)\le t_1$.

Case 2: Schedule job J_j to be finished after r on machine M_1. In this case, when the scenario $\gamma=0$ with probability ζ_0 happens, the completion time of job J_j is $t_1+v_1+p_j(1+a(t_1+v_1))$ since the machine efficiency is not improved during the scheduling horizon; however, when scenario $\gamma=1,\ldots,s$, with probability ζ_γ happens, the completion time of job J_j is $r+\kappa+q_1+p_j(1+aq_1)$ if $1\le\gamma\le\kappa$, and $r+\gamma+q_1+p_j(1+aq_1)$ otherwise since the machine maintenance takes κ time units. Hence, the expected completion time of job J_j is $\zeta_0(t_1+v_1+p_j(1+a(t_1+v_1)))+\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+q_1+p_j(1+aq_1))$. Therefore, if $t_1+v_1+p_j(1+a(t_1+v_1))>r$, which ensures that J_j will be finished after r when $\gamma=0$ happens, we include the new state $(u_1,u_2,v_1+p_j(1+a(t_1+v_1)),q_1+p_j(1+aq_1),f+\zeta_0(t_1+v_1+p_j(1+a(t_1+v_1)))+\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+q_1+p_j(1+aq_1)))$ in $\mathcal{F}_{(j,t_1)}$.

Case 3: Schedule job J_j on machine M_2. In this case, the contribution of job J_j to the objective function is $u_2+p_j(1+au_2)$. Hence, we include the new state $(u_1,u_2+p_j(1+au_2),v_1,q_1,f+u_2+p_j(1+au_2))$ in $\mathcal{F}_{(j,t_1)}$.

Note that the process to construct $\mathcal{F}_{(j,t_1)}$ may generate more than a single state that does not lead to a complete optimal schedule. The following result shows how to reduce the set $\mathcal{F}_{(j,t_1)}$.

Lemma 4.1. *For any two states (u_1,u_2,v_1,q_1,f) and (u_1',u_2',v_1',q_1',f') in $\mathcal{F}_{(j,t_1)}$ with $u_1\le u_1'$, $u_2\le u_2'$, $v_1\le v_1'$, $q_1\le q_1'$, and $f\le f'$, we can eliminate the latter state.*

Proof. This is due to the fact that for any extension for the partial schedule corresponding to the state (u_1',u_2',v_1',q_1',f') to a solution for the complete problem, the corresponding extension for the partial schedule corresponding to the state (u_1,u_2,v_1,q_1,f) yields a feasible solution whose f value is at least as good as that of the former. □

We summarize the results of the above analysis in the following solution algorithm for the problem $P2,1|nr-a,M,DE,PDR|\ E\left(\sum_{j=1}^{n}C_j\right)$ with $\zeta_0\neq 0$.

Algorithm *NRMDPDDP*

Step 1. [*Preprocessing*] Re-index the jobs in the SPT order.
Step 2. [*Initialization*] Set $\tau=\lceil\frac{r-p_{\max}}{1+ap_{\max}}\rceil$ and $\mathcal{F}_{(0,t_1)}=\{(0,0,0,0,0)\}$ for each $t_1=\tau,\ldots,r$.
Step 3. [*Generation*]
For $t_1=\tau$ to r do
  For $j=1$ to n do
    Set $\mathcal{F}_{(j,t_1)}=\varnothing$
    For each $(u_1,u_2,v_1,q_1,f)\in\mathcal{F}_{(j-1,t_1)}$ do
      /* Schedule job J_j before t_1 on machine M_1
      If $u_1+p_j(1+au_1)\le t_1$, then
        Set $\mathcal{F}_{(j,t_1)}\leftarrow\mathcal{F}_{(j,t_1)}\cup\{(u_1+p_j(1+au_1),u_2,v_1,q_1,f+u_1+p_j(1+au_1))\}$;
      Endif
      /* Schedule job J_j to be finished after r on machine M_1
      If $t_1+v_1+p_j(1+a(t_1+v_1))>r$, then
        Set $\mathcal{F}_{(j,t_1)}\leftarrow\mathcal{F}_{(j,t_1)}\cup\{(u_1,u_2,v_1+p_j(1+a(t_1+v_1)),q_1+p_j(1+aq_1),f+$
          $\zeta_0(t_1+v_1+p_j(1+a(t_1+v_1)))+\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+q_1+p_j(1+aq_1)))\}$;
      Endif
      /* Schedule job J_j on machine M_2
      Set $\mathcal{F}_{(j,t_1)}\leftarrow\mathcal{F}_{(j,t_1)}\cup\{(u_1,u_2+p_j(1+au_2),v_1,q_1,f+u_2+p_j(1+au_2))\}$;
    Endfor
    [*Elimination*] /* Update set $\mathcal{F}_{(j,t_1)}$ */
    For any two states (u_1,u_2,v_1,q_1,f) and (u_1',u_2',v_1',q_1',f') in $\mathcal{F}_{(j,t_1)}$ with $u_1\le u_1'$,
    $u_2\le u_2'$, $v_1\le v_1'$, $q_1\le q_1'$, and $f_1\le f_1'$, eliminate the latter state from set $\mathcal{F}_{(j,t_1)}$;
  Endfor
Endfor
Step 4. [*Result*] The optimal solution value is given by $\min\{f\,|\,(u_1,u_2,v_1,q_1,f)\in\mathcal{F}_{(n,t_1)},\ t_1=\kappa,\ldots,r\}$ and the optimal solution can be found by backtracking.

Theorem 4.2. *Algorithm NRMDPDDP solves the problem $P2,1|nr-a,M,DE,PDR|\ E(\sum_{j=1}^{n}C_j)$ with $\zeta_0\neq 0$ in $O(nr^2TP^3)$ time.*

Proof. The optimality of algorithm *NRMDPDDP* follows directly from Lemma 3.3 and the above analysis. We now analyze its time complexity. Step 1 is a sorting procedure, which takes $O(n\log n)$ time. In Step 3, an upper bound on the number of the vectors $\{u_1,u_2,v_1,q_1\}$ is rTP^3 because, by Lemma 3.1, there are at most r possible values for u_1 and at most TP possible values for each of u_2, v_1, and q_1,

Please cite this article as: Yin Y, et al. Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.07.006

respectively. Thus, for each given t_1, before each iteration j, there are at most rTP^3 possible states $(u_1,u_2,v_1,q_1,f)\in\mathcal{F}_{(j-1,t_1)}$ because the number of the vectors $\{u_1,u_2,v_1,q_1,f\}$ is upper-bounded by rTP^3 due to the elimination rules. In iteration j, there are at most three new states generated from each state in $\mathcal{F}_{(j-1,t_1)}$ for each candidate job. Therefore, there are at most $3rTP^3$ new states generated in $\mathcal{F}_{(j,t_1)}$. Thus, the construction of $\mathcal{F}_{(j,t_1)}$ requires $O(rTP^3)$ time, which is also the time required for the elimination process. After nr iterations, Step 3 can be executed in $O(nr^2TP^3)$ time, as required. Step 4 takes $O(r^2TP^3)$ time. Therefore, the overall time complexity of the algorithm is indeed $O(nr^2TP^3)$. □

If $\zeta_0=0$, i.e., the anticipated disruption will definitely happen although the duration is uncertain, the time complexity of algorithm *NRMDPDDP* can be reduced to $O(nrTP^2)$ since the parameters t_1 and v_1 can be dropped in the state vector. We give a formal description of the algorithm for this case in the following, where $\mathcal{F}_{(j)}$ is the state space generated in the jth iteration.

Algorithm *NRMDPDDP0*

Step 1. [*Preprocessing*] Re-index the jobs in the SPT order.
Step 2. [*Initialization*] Set $\mathcal{F}_{(0)}=\{(0,0,0,0)\}$.
Step 3. [*Generation*]
For $j=1$ to n do
Set $\mathcal{F}_{(j)}=\varnothing$
[*State Generation*]
For each $(u_1,u_2,q_1,f)\in\mathcal{F}_{(j-1)}$ do
/* Schedule job J_j before r on machine M_1
If $u_1+p_j(1+au_1)\le r$, then
Set $\mathcal{F}_{(j)}\leftarrow\mathcal{F}_{(j)}\cup\{(u_1+p_j(1+au_1),u_2,q_1,f+u_1+p_j(1+au_1))\}$;
Endif
/* Schedule job J_j to be finished after r on machine M_1
Set $\mathcal{F}_{(j)}\leftarrow\mathcal{F}_{(j)}\cup\{(u_1,u_2,q_1+p_j(1+aq_1),f+\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+q_1+p_j(1+aq_1)))\}$;
/* Schedule job J_j on machine M_2
Set $\mathcal{F}_{(j)}\leftarrow\mathcal{F}_{(j)}\cup\{(u_1,u_2+p_j(1+au_2),q_1,f+u_2+p_j(1+au_2))\}$;
Endfor
[*Elimination*] /* Update set $\mathcal{F}_{(j)}$ */
The same as in algorithm *NRMDPDDP*;
Endfor
Step 4. [*Result*] The optimal solution value is given by $\min\{f\mid(u_1,u_2,q_1,f)\in\mathcal{F}_{(n)}\}$ and the optimal solution can be found by backtracking.

Hence, we conclude with the following result.

Theorem 4.3. *Algorithm NRMDPDDP0 solves the problem* $P2,1\mid nr-a,M,DE,PDR\mid E\left(\sum_{j=1}^{n}C_j\right)$ *with* $\zeta_0=0$ *in* $O\left(nrTP^2\right)$ *time.*

Note that the idea of algorithm *NRMDPDDP* can be applied for the general case with m machines and any given ι, in which we should include the new variables t_i to denote the starting time of the first job scheduled on machine M_i that will finish after r in the final optimal schedule when the disruption does not happen on machine M_i, and v_i and q_i to denote the total actual processing time of the jobs finished after r on machine M_i for the cases where the disruption does not happen and where the anticipated disruption does happen on machine M_i, respectively, for all $i=2,\ldots,\iota$. We conclude with the following result.

Theorem 4.4. *The problem* $Pm,\iota\mid nr-a,M,DE,PDR\mid E\left(\sum_{j=1}^{n}C_j\right)$ *can be solved in* $O(nr^{2\iota}TP^{m+\iota})$ *time if* $\zeta_0\neq0$ *and in* $O(nr^{\iota}TP^{m})$ *time otherwise.*

4.1.2. The problem $Pm,\iota\mid nr-a,NM,DE,PDR\mid E\left(\sum_{j=1}^{n}C_j\right)$

This subsection focuses on the case of not performing machine maintenance. For simplicity, we only analyze the two-machine case with $\iota=1$ and $\zeta_0\neq0$ in detail, and develop a forward dynamic programming algorithm *NRNMDPDDP* for solving it.

The idea of algorithm *NRNMDPDDP* is analogous to that of *NRMDPDDP*. Let t_1 and τ be defined as before. For each given t_1, the algorithm consists of $n+1$ phases and in each phase j, $j=0,1,\ldots,n$, a state space $\mathcal{L}_{(j,t_1)}$ is generated. Any state in $\mathcal{L}_{(j,t_1)}$ is a vector $(u_1,u_2,v_1,\mathfrak{q}_1,f)$ that encodes a feasible partial schedule for the jobs $\{J_1,\ldots,J_j\}$, where the variables u_1, u_2, v_1, and f are defined as before, and $\mathfrak{q}_1=(q_1^1,\ldots,q_1^s)$ is a vector in which q_1^γ, $\gamma=1,\ldots,s$, denotes the total actual processing times of the jobs finished after r on machine M_1 under the scenario γ. The state spaces $\mathcal{L}_{(j,t_1)}$, $j=0,1,\ldots,n$, $t_1=\kappa,\ldots,r$, are constructed iteratively. The initial space $\mathcal{F}_{(0,t_1)}$ for each $t_1=\kappa,\ldots,r$ contains $(0,0,0,0,0)$ with $=(\underbrace{0,\ldots,0}_{s})$ as its only element. For each given t_1, in the jth phase, $j=1,\ldots,n$, we build a state by adding a single job J_j to a previous state, if it is possible for the given state. To add job J_j to a state $(u_1,u_2,v_1,\mathfrak{q}_1,f)\in\mathcal{L}_{(j-1,t_1)}$, there are three cases analogous to those in algorithm *NRMDPDDP* to consider. The procedures for Cases 1 and 3 are the same as those in algorithm *NRMDPDDP*. We illustrate Case 2 here, i.e., scheduling job J_j to be finished after r on machine M_1. In this case, under the scenario $\gamma=0$ with probability ζ_0, the completion time of job J_j is $t_1+v_1+p_j(1+a(t_1+v_1))$; however, when scenario $\gamma=1,\ldots,s$, with probability ζ_γ happens, the completion time of job J_j is $r+\gamma+q_1^\gamma+p_j(1+a(r+\gamma+q_1^\gamma))$. Hence, the expected completion time of job J_j is $\zeta_0(t_1+v_1+\ p_j(1+a(t_1+v_1)))+\sum_{\gamma=1}^{s}\zeta_\gamma(r+\gamma+q_1^\gamma+p_j(1+a(r+\gamma+q_1^\gamma))$. Therefore, if $t_1+v_1+p_j(1+a(t_1+v_1))>r$, which ensures that J_j will be finished after r when $\gamma=0$ happens, we include the new state $(u_1,u_2,v_1+p_j(1+a(t_1+v_1)),\overline{\mathfrak{q}}_1,f+\zeta_0(t_1+v_1+p_j(1+a(t_1+v_1)))+\sum_{\gamma=1}^{s}\zeta_\gamma(r+\gamma+q_1^\gamma+p_j(1+a(r+\gamma+q_1^\gamma))))$ in $\mathcal{L}_{(j,t_1)}$, where $\overline{\mathfrak{q}}_1=(\overline{q}_1^1,\ldots,\overline{q}_1^s)$ with $\overline{q}_1^\gamma=q_1^\gamma+p_j(1+a(r+\gamma+q_1^\gamma))$, $\gamma=1,\ldots,s$.

Please cite this article as: Yin Y, et al. Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.07.006

Before presenting the dynamic programming algorithm in detail, a similar elimination property can be derived to reduce the state space.

Lemma 4.5. *For any two states* $(u_1, u_2, v_1, \mathfrak{q}_1, f)$ *and* $(u_1', u_2', v_1', \mathfrak{q}_1', f')$ *in* $\mathcal{L}_{(j,t_1)}$ *with* $u_1 \le u_1'$, $u_2 \le u_2'$, $v_1 \le v_1'$, $\mathfrak{q}_1 \le \mathfrak{q}_1'$, *and* $f \le f'$, *we can eliminate the latter state, where* $\mathfrak{q}_1 \le \mathfrak{q}_1'$ *means that* $q_1^\gamma \le q_1'^\gamma$ *for all* $\gamma = 1, \ldots, s$.

Proof. The proof is analogous to that of Lemma 4.1. □

We summarize the results of the above analysis in the following solution algorithm for the problem $P2, 1 \mid nr-a, NM, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$ with $\zeta_0 \neq 0$.

Algorithm *NRNMDPDDP*

Step 1. [*Preprocessing*] Re-index the jobs in the SPT order.
Step 2. [*Initialization*] Set $\tau = \lceil \frac{r - p_{\max}}{1 + ap_{\max}} \rceil$ and $\mathcal{L}_{(0,t_1)} = \{(0, 0, , 0, 0)\}$ for each $t_1 = \tau, \ldots, r$, where $= (\underbrace{0, \ldots, 0}_{s})$.

Step 3. [*Generation*]
For $t_1 = \tau$ to r do
For j=1 to n do
Set $\mathcal{L}_{(j,t_1)} = \varnothing$
For each $(u_1, u_2, v_1, \mathfrak{q}_1, f) \in \mathcal{L}_{(j-1,t_1)}$ do
/* Schedule job J_j before t_1 on machine M_1
If $u_1 + p_j(1+au_1) \le t_1$, then
Set $\mathcal{L}_{(j,t_1)} \leftarrow \mathcal{L}_{(j,t_1)} \cup \{(u_1 + p_j(1+au_1), u_2, v_1, \mathfrak{q}_1, f + u_1 + p_j(1+au_1))\}$;
Endif
/* Schedule job J_j to be finished after r on machine M_1
If $t_1 + v_1 + p_j(1+a(t_1+v_1)) > r$, then
Set $\mathcal{L}_{(j,t_1)} \leftarrow \mathcal{L}_{(j,t_1)} \cup \{(u_1, u_2, v_1 + p_j(1+a(t_1+v_1)), \overline{\mathfrak{q}}_1, f +$
$\zeta_0(t_1 + v_1 + p_j(1+a(t_1+v_1))) + \sum_{\gamma=1}^{s} \zeta_\gamma(r+\gamma+q_1^\gamma + p_j(1+a(r+\gamma+q_1^\gamma))))\}$,
where $\overline{\mathfrak{q}}_1 = (\overline{q}_1^1, \ldots, \overline{q}_1^s)$ with $\overline{q}_1^\gamma = q_1^\gamma + p_j(1+a(r+\gamma+q_1^\gamma))$, $\gamma = 1, \ldots, s$;
Endif
/* Schedule job J_j on machine M_2
Set $\mathcal{L}_{(j,t_1)} \leftarrow \mathcal{L}_{(j,t_1)} \cup \{(u_1, u_2 + p_j(1+au_2), v_1, \mathfrak{q}_1, f + u_2 + p_j(1+au_2))\}$;
Endfor
[*Elimination*] /* Update set $\mathcal{L}_{(j,t_1)}$ */
For any two states $(u_1, u_2, v_1, \mathfrak{q}_1, f)$ and $(u_1', u_2', v_1', \mathfrak{q}_1', f')$ in $\mathcal{L}_{(j,t_1)}$ with $u_1 \le u_1'$,
$u_2 \le u_2'$, $v_1 \le v_1'$, $\mathfrak{q}_1 \le \mathfrak{q}_1'$, and $f \le f'$, eliminate the latter state from set $\mathcal{L}_{(j,t_1)}$;
Endfor
Endfor

Step 4. [*Result*] The optimal solution value is given by $\min\{f \mid (u_1, u_2, v_1, \mathfrak{q}_1, f) \in \mathcal{L}_{(n,t_1)}, t_1 = \kappa, \ldots, r\}$ and the optimal solution can be found by backtracking.

Theorem 4.6. *Algorithm NRNMDPDDP solves the problem* $P2, 1 \mid nr-a, NM, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$ *with* $\zeta_0 \neq 0$ *in* $O(nr^2\, \mathcal{TP}^{s+1} TP)$ *time.*

Proof. The proof is analogous to that of Theorem 4.2, the only difference being that the number of different combinations of $\{u_1, u_2, v_1, \mathfrak{q}_1\}$ is upper-bounded by $r\mathcal{TP}^{s+1}TP$ because, by Lemma 3.1, there are at most r possible values for u_1, at most TP possible values for u_2, and at most $\mathcal{TP}$ values for each of v_1 and q_1^γ, $\gamma = 1, \ldots, s$, respectively. □

Analogous to the analysis in Section 4.1.1, algorithm *NRNMDPDDP* can be generalized to solving the general problem $Pm, \iota \mid nr-a, NM, DE, PDR \mid E\left(\sum_{j=1}^{n} C_j\right)$ with $\zeta_0 \neq 0$ or $\zeta_0 = 0$, and the following result holds.

Theorem 4.7. *The problem* $Pm, \iota \mid nr-a, NM, DE, PDR \mid E\left(\sum_{j=1}^{n} C_j\right)$ *can be solved in* $O\left(nr^{2\iota}\mathcal{TP}^{(s+1)\iota}TP^{m-\iota}\right)$ *time if* $\zeta_0 \neq 0$ *and in* $O\left(nr^{\iota}\mathcal{TP}^{s\iota}TP^{m-\iota}\right)$ *time otherwise.*

4.2. FPTASs for the case where $\iota = 1$ and $\zeta_0 = 0$

In this subsection we first describe how to convert algorithm ıNRMDPDDP0 into an FPTAS and then briefly discuss how to generalize the result to the case with m machines, $\iota = 1$ and $\zeta_0 = 0$. To do this, we borrow the idea from Hall and Potts [7] to partition the state space into boxes, which originates from the interval partitioning approach suggested by Sahni [24]. Specifically, to convert algorithm *NRMDPDDP*0 into an FPTAS, we partition the state space $\mathcal{F}_{(j)}$, $j = 1, \ldots, n$ into three-dimensional boxes and approximate the solution by retaining only one state within any box. We formally describe the procedure as follows:

Algorithm $NRMDPDAA_\varepsilon$

Step 1. [*Preprocessing*] Re-index the jobs in the SPT order.
Step 2. [*Initialization*] Set $\mathcal{F}_{(0)} = \{(0, 0, 0, 0)\}$.
Step 3. [*Generation*]

Please cite this article as: Yin Y, et al. Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.07.006

For $j=1$ to n do
Set $\mathcal{F}_{(j)}=\varnothing$
[*State Generation*] /* Generate $\mathcal{F}_{(j)}$ from $\mathcal{F}_{(j-1)}$ */
The same as that in algorithm *NRMDPDDP0*;
[*Labeling*] /* Attach a label to each state in $\mathcal{F}_{(j)}$ */
For each state $(u_1,u_2,q_1,f)\in\mathcal{F}_{(j)}$, attach the label $(\Delta(u_2),\Delta(q_1),\Delta(f))$ to it, where the function Δ is defined as $\Delta(x)=k$ in which x satisfies $\delta^k\le x\le\delta^{(k+1)}$ and $\delta=1+\frac{\varepsilon}{2(1+\varepsilon)n}$;
[*Elimination*] /* Update set $\mathcal{F}_{(j)}$ */
(1) For any two states (u_1,u_2,q_1,f) and (u_1',u_2,q_1,f) in $\mathcal{F}_{(j)}$ with the same label and $u_1\le u_1'$, eliminate the latter state from set $\mathcal{F}_{(j)}$;
(2) For any two states (u_1,u_2,q_1,f) and (u_1,u_2',q_1',f') in $\mathcal{F}_{(j)}$ with $u_2\le u_2'$, $q_1\le q_1'$, and $f_1\le f_1'$, eliminate the latter state from set $\mathcal{F}_{(j)}$;
Endfor

Step 4. [*Result*] The approximate solution value is given by $\min\{f\mid(u_1,u_2,q_1,f)\in\mathcal{F}_{(n)}\}$ and the approximate solution can be found by backtracking.

Lemma 4.8. *For any eliminated state $(u_1,u_2,q_1,f)\in\mathcal{F}_{(j)}$, there exists a state $(\widetilde{u_1},\widetilde{u_2},\widetilde{q_1},\widetilde{f})$ such that $\widetilde{u_1}\le u_1$, $\widetilde{u_2}\le\delta^j u_2$, $\widetilde{q_1}\le\delta^j q_1$, and $\widetilde{f}\le\delta^j f$.*

Proof. Refer to Appendix for details. □

Theorem 4.9. *For any $\varepsilon>0$ and an optimal solution value f^*, algorithm NRMDPDAA$_\varepsilon$ finds in $O\left(\frac{n^7L^3}{\varepsilon^3}\right)$ time a solution value $\widetilde{f}$ such that $\widetilde{f}\le(1+\varepsilon)f^*$.*

Proof. Let $(u_1^*,u_2^*,q_1^*,f^*)\in\mathcal{F}_{(n)}$ be a state corresponding to the optimal solution value f^*. By the proof of Lemma 4.8, for (u_1^*,u_2^*,q_1^*,f^*), there exists a non-eliminated state $(\widetilde{u_1},\widetilde{u_2},\widetilde{q_1},\widetilde{f})$ such that $\widetilde{f}\le\delta^n f^*$. It follows from $(1+\frac{x}{n})^n\le1+2x$, for any $0\le x\le1$, that $\left(1+\frac{\varepsilon}{2(1+\varepsilon)n}\right)^n\le1+\frac{\varepsilon}{1+\varepsilon}\le1+\varepsilon$, so $\widetilde{f}\le\delta^n f^*\le(1+\varepsilon)f^*$.

For the time complexity of algorithm NRMDPDAA$_\varepsilon$, Step 1 requires $O(1)$ time. Note that there are at most TP possible values for each of u_2 and q_1, respectively, and at most nTP possible values for f. Therefore, the number of possible values of $\Delta(u_2)$ and $\Delta(q_1)$ are given by

$$\lceil\log_\delta TP\rceil=\lceil\ln TP/\ln\delta\rceil\le\lceil(1+2n(1+\varepsilon)/\varepsilon)\ln TP\rceil\le\left\lceil(1+2n(1+\varepsilon)/\varepsilon)\left(\sum_{k=1}^{n}\ln(1+ap_k)-\ln a\right)\right\rceil,$$

where the first inequality is obtained from the well-known inequality $\ln x\ge(x-1)/x$ for all $x\ge1$. Similarly, the number of possible values of $\Delta(f)$ is given by

$$\lceil\log_\delta nTP\rceil\le\left\lceil(1+2n(1+\varepsilon)/\varepsilon)\left(\sum_{k=1}^{n}\ln(1+ap_k)+\ln n-\ln a\right)\right\rceil.$$

Thus, the total number of different boxes at the beginning of each iteration, which equals the number of different states at the beginning of each iteration since at most one state is retained for each possible label after the elimination process, is at most $O\left(\frac{n^6L^3}{\varepsilon^3}\right)$. In each iteration j, there are at most three new states generated from each state in $\mathcal{F}_{(j-1)}$. Thus, the number of new states generated is at most $3\times O\left(\lceil(1+2n(1+\varepsilon)/\varepsilon)(\sum_{k=1}^{n}\ln(1+ap_k)-\ln a)\rceil^2\lceil(1+2n(1+\varepsilon)/\varepsilon)(\sum_{k=1}^{n}\ln(1+ap_k)+\ln n-\ln a)\rceil\right)$, which equals $3\times O\left(\frac{n^6L^3}{\varepsilon^3}\right)$. Thus, the construction of $\mathcal{F}_{(j)}$ requires $O\left(\frac{n^6L^3}{\varepsilon^3}\right)$ time, which is also the time required for the elimination process. After n iterations, the total number of different boxes is at most $O\left(\frac{n^7L^3}{\varepsilon^3}\right)$, which is also the overall time complexity of algorithm NRMDPDAA$_\varepsilon$. □

Note that the idea of algorithm NRMDPDAA$_\varepsilon$ can be generalized to the general case with m machines, $\iota=1$ and $\zeta_0=0$, in which we should include the new variables u_i to denote the total actual processing time of the jobs scheduled on machine M_i for all $i=3,\ldots,m$ in the state vector. We conclude with the following result.

Theorem 4.10. *The problem $Pm,1\mid nr-a,M,DE,PDR\mid E(\sum_{j=1}^{n}C_j)$ with $\zeta_0=0$ admits an FPTAS that runs in $O\left(\frac{n^{2m+3}L^{m+1}}{\varepsilon^{m+1}}\right)$ time.*

In a similar way, the following result holds.

Theorem 4.11. *The problem $Pm,1\mid nr-a,NM,DE,PDR\mid E(\sum_{j=1}^{n}C_j)$ with $\zeta_0=0$ admits an FPTAS that runs in $O\left(\frac{n^{2(s-1)\iota+2m+3}L^{(s-1)\iota+m+1}}{\varepsilon^{(s-1)\iota+m+1}}\right)$ time.*

Remark 4.12. It is well known that FPTAS is the strongest type of approximation result for an *NP*-hard problem. Theorems 4.10 and 4.11 indicate that in the non-resumable case, no matter whether performing machine maintenance or not, the problem admits an FPTAS when m is fixed, $\iota=1$, and $\zeta_0=0$. However, the time complexity of the FPTAS may pose a challenge to solving the problem in practice, so future research should design fast and efficient constant ratio approximation algorithms for the problems.

5. The resumable case

It is worth noting that the proof given in Levin et al. [17] for the problem $P,1\mid nr-a\mid\sum_{j=1}^{n}C_j$ can be applied for the resumable case, implying that our problem when both m and ι are arbitrary, denoted as $P,\iota\mid r-a,M/NM,DE,PDR\mid E(\sum_{j=1}^{n}C_j)$, is also *NP*-hard in the strong sense. In addition, since even for the case without deteriorating effect on two identical parallel machines with $\iota=2$ and $\zeta_s=1$, the

problem is *NP*-hard [16], our problem with a fixed m is *NP*-hard, too, when $\iota \geq 2$. In the following we focus on the case with a fixed m under the resumable availability constraint, and design a pseudo-polynomial-time solution algorithm for each of the cases of performing and not performing machine maintenance, establishing that it is *NP*-hard in the ordinary sense. However, the question as to whether or not the problem with $\iota = 1$ is *NP*-hard remains open.

Before developing the algorithm, we note that scheduling the jobs on each machine in the SPT order is optimal for the case without the deteriorating effect on m identical parallel machines with $\iota = m$ under the resumable availability constraint (see Theorem 4 in Lee and Yang [16]). However, this result does not hold for our problem even for the single-machine case, as shown in the following example.

Example 5.1. Let $n = 3, m = 1$; $p_1 = 10, p_2 = 50, p_3 = 90$; $a = 0.1, r = 40, \kappa = s = 5$, and $\zeta_s = 1$.

For the case of performing maintenance, consider a solution obtained by splitting job J_2 into two pieces J_2' and J_2'' with normal processing times 15 and 35, and scheduling jobs J_1, J_2', J_2'', and J_3 in the intervals [0,10], [10,40], [50,85], and [85,490], respectively. It fulfils the property that the jobs follow the SPT order and yields an objective value of 585. However, a better solution is obtained by splitting job J_2 into two pieces J_2' and J_2'' with normal processing times 40 and 10, and scheduling jobs J_2', J_2'', J_1, and J_3 in the intervals [0,40], [50,60], [60,80], and [80,440], respectively, which yields a smaller objective value of 580.

Similarly, for the case of not performing maintenance, the solution in which the jobs follow the SPT order yields an objective value of 2700. However, a better solution with objective value of 2500 is obtained by splitting job J_2 into two pieces J_2' and J_2'' with normal processing times 40 and 10, respectively, and scheduling the jobs in the following order: J_2', J_2'', J_1, and J_3.

5.1. The problem $Pm, \iota \mid r-a, M, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$

This subsection focuses on the case of performing machine maintenance. Again, we only analyze the two-machine case with $\iota = 1$ and $\zeta_0 \neq 0$ in detail, and develop a forward dynamic programming algorithm *RMDPDDP* for solving it.

The idea of algorithm *RMDPDDP* is analogous to that of *NRMDPDDP*, except that we need to enumerate all the possible disrupted jobs that start processing no later than r but finish processing after r on machine M_1 by Lemma 3.3 and Example 5.1. Thus we have to add a new parameter k_1 to denote that the disrupted job on machine M_1 is job J_{k_1} in the final optimal schedule. For each fixed k_1, let τ_1 denote the starting time of job J_{k_1} on machine M_1 in the final optimal schedule, which must satisfy $\tau_1 + p_{k_1}(1+a\tau_1) > r$, implying that τ_1 ranges from χ_{k_1} to r, where $\chi_{k_1} = \lceil \frac{r-p_{k_1}}{1+ap_{k_1}} \rceil$. For any feasible combination of k_1 and τ_1, we set $\iota_{(k_1,\tau_1)} = (r-\tau_1)/(1+a\tau_1)$, which denotes the normal processing part of job J_{k_1} processed before r when the disruption does happen, and re-index J_{k_1} as J_{n+1} and J_j as J_{j-1} for $j = k_1+1, \ldots, n$. For each given feasible combination of k_1 and τ_1, this algorithm consists of n phases and in each phase j, $j = 0, 1, \ldots, n-1$, a state space $\mathcal{F}_{(j,k_1,\tau_1)}$ is generated. Any state in $\mathcal{F}_{(j,k_1,\tau_1)}$ is a vector (u_1, u_2, v_1, q_1, f) that encodes a feasible partial schedule for the jobs $\{J_1, \ldots, J_j\}$, where the variables u_1, u_2, v_1, q_1, and f are defined as before. The state spaces $\mathcal{F}_{(j,k_1,\tau_1)}$, $j = 0, 1, \ldots, n-1, k_1 = 1, \ldots, n, \tau_1 = \chi_{k_1}, \ldots, r$, are constructed iteratively, where the initial space $\mathcal{F}_{(0,k_1,\tau_1)}$ for $k_1 = 1, \ldots, n$ and $\tau_1 = \chi_{k_1}, \ldots, r$, contains $(0, 0, p_{n+1}(1+a\tau_1), p_{n+1} - \iota_{(k_1,\tau_1)}, \zeta_0(\tau_1 + p_{n+1}(1+a\tau_1)) + \sum_{\gamma=1}^{s} \zeta_\gamma(r + \max\{\gamma, \kappa\} + p_{n+1} - \iota_{(k_1,\tau_1)}))$ as its only element. For any possible combination of k_1 and τ_1, in the jth phase, $j = 1, \ldots, n-1$, we build a state by adding a single job J_j to a previous state in an analogous way to that in algorithm *NRMDPDDP*, if it is possible for the given state. Note also that Lemma 4.1 still holds for this case.

The solution procedure for the problem $P2, 1 \mid r-a, M, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$ with $\zeta_0 \neq 0$ can be formally described as follows.

Algorithm *RMDPDDP*

Step 1. [*Preprocessing*] Re-index the jobs in the SPT order.

Step 2. [*Initialization*] For any $k_1 = 1, \ldots, n$, set $\chi_{k_1} = \lceil \frac{r-p_{k_1}}{1+ap_{k_1}} \rceil$. For any $k_1 = 1, \ldots, n$ and $t_1 = \chi_{k_1}, \ldots, r$, set $\iota_{(k_1,\tau_1)} = (r-\tau_1)/(1+a\tau_1)$ and let $\mathcal{F}_{(0,k_1,t_1)} = \{(0, 0, p_{k_1}(1+a\tau_1), p_{k_1} - \iota_{(k_1,\tau_1)}, \zeta_0(\tau_1 + p_{k_1}(1+a\tau_1)) + \sum_{\gamma=1}^{\kappa} \zeta_\gamma(r + \max\{\gamma, \kappa\} + p_{k_1} - \iota_{(k_1,\tau_1)}))\}$.

Step 3. [*Generation*]

```
For k1 = 1 to n do
  For the original SPT order do: re-index J_{k1} as J_{n+1} and J_j as J_{j-1} for j = k1+1, ..., n;
  For τ1 = χ_{k1} to r do
    For j = 1 to n-1 do
      Set F_{(j,k1,τ1)} = ∅
      For each (u1, u2, v1, q1, f) ∈ F_{(j-1,k1,τ1)} do
        /* Schedule job J_j before τ1 on machine M1
        If u1 + p_j(1+au1) ≤ τ1, then
          Set F_{(j,k1,τ1)} ← F_{(j,k1,τ1)} ∪ {(u1 + p_j(1+au1), u2, v1, q1, f + u1 + p_j(1+au1))};
        Endif
        /* Schedule job J_j to be finished after r on machine M1
        Set F_{(j,k1,τ1)} ← F_{(j,k1,τ1)} ∪ {(u1, u2, v1 + p_j(1+a(τ1+v1)), q1 + p_j(1+aq1), f + ζ0(τ1 + v1 + p_j(1+a(τ1+v1)))
        + Σ_{γ=1}^{s} ζ_γ(r + max{γ, κ} + q1 + p_j(1+aq1)))};
        /* Schedule job J_j on machine M2
        Set F_{(j,k1,τ1)} ← F_{(j,k1,τ1)} ∪ {(u1, u2 + p_j(1+au2), v1, q1, f + u2 + p_j(1+au2))};
      Endfor
      [Elimination] /* Update set F_{(j,k1,τ1)} */
      The same as that in algorithm NRMDPDDP;
  Endfor
Endfor
```

Please cite this article as: Yin Y, et al. Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.07.006

Step 4. [*Result*] The optimal solution value is given by $\min\{f \mid (u_1, u_2, v_1, q_1, f) \in \mathcal{F}_{(n-1,k_1,\tau_1)},\ k_1 = 1, \ldots, n, \tau_1 = \chi_{k_1}, \ldots, r\}$ and the optimal solution can be found by backtracking.

Theorem 5.2. *Algorithm RMDPDDP solves the problem* $P2, 1 \mid r-a, M, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$ *with* $\zeta_0 \neq 0$ *in* $O(n^2 r^2 T P^3)$ *time.*

Proof. The proof is analogues to that of Theorem 4.2. □

Analogous to the analysis in Subsection 4.1.1, algorithm *RMDPDDP* can be applied for solving the general problem $Pm, \iota \mid r-a, M, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$ with slightly modifications, and we conclude the following result.

Theorem 5.3. *The problem* $Pm, \iota \mid r-a, M, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$ *can be solved in* $O(n^{\iota+1} r^{2\iota} T P^{m+\iota})$ *time if* $\zeta_0 \neq 0$ *and in* $O(n^{\iota+1} r^{2\iota} T P^{m})$ *time otherwise.*

5.2. The problem $Pm, \iota \mid r-a, NM, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$

This subsection focuses on the case of not performing machine maintenance. The idea of the algorithm for this case with $\zeta_0 \neq 0$ is analogous to those of algorithms NRNMDPDDP and *RMDPDDP*. Compared with algorithm *RMDPDDP*, we need new parameters q_i^{γ}, $i = 1, \ldots, \iota$, $\gamma = 1, \ldots, s$, to denote the total actual processing times of the jobs finished after r on machine M_i under the scenario γ in the state vector. Thus, combining Theorems 4.7 and 5.3, we obtain the following result.

Theorem 5.4. *The problem* $Pm, \iota \mid r-a, NM, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$ *can be solved in* $O(n^{\iota+1} r^{2\iota}\ \mathcal{TP}^{(s+1)\iota}\ TP^{m-\iota})$ *time if* $\zeta_0 \neq 0$ *and in* $O(n^{\iota+1} r^{2\iota}\ \mathcal{TP}^{s\iota} TP^{m-\iota})$ *time otherwise.*

6. Conclusions

This paper addresses parallel-machine scheduling of deteriorating jobs with potential machine disruptions. In such a case, the job processing times will deteriorate over time and some of the machines may become unavailable for some periods of time due to the potential machine disruptions. The disruption starting time is often known (should it happen) in advance, yet the disruption duration that will last for a certain duration with a certain probability is unknown until it happens. We consider the cases of performing maintenance immediately on the disrupted machines when a disruption occurs and not performing machine maintenance, where performing machine maintenance will improve the efficiency of a machine, making it return to its original state of efficiency. In each case, the objective is to find an optimal schedule to minimize the expected total completion time of the jobs for both the non-resumable and resumable cases. We summarize the major results of this paper in Table 1.

It is worth noting that performing maintenance on the disrupted machines when a disruption occurs will decrease the expected total completion time of the jobs at the expense of the maintenance cost. When this cost is included in the objective function, it is not difficult to make a decision whether or not to perform machine maintenance immediately on the disrupted machines when a disruption occurs by using our algorithms.

For future research, we suggest several interesting topics as follows:

- Ascertain the computational complexity status of the problem $Pm, 1 \mid r-a, M/NM, DE, PMD \mid E(\sum_{j=1}^{n} C_j)$.
- Design efficient constant ratio approximation algorithms for the problems under consideration.
- Extend our model to the case where the objective is to determine the optimal maintenance starting time and optimal schedule to minimize the expected total completion time of the jobs.

Table 1
Summary of results.

Problem	Complexity	Reference
$P, \iota \mid r/nr-a, M/NM, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$	*NP*-hard in the strong sense	Levin et al. [17]
$Pm, \iota \mid nr-a, M, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$	*NP*-hard in the ordinary sense	Lee and Liman [14]
	$O(nr^{2\iota} TP^{m+\iota})$: DP if $\zeta_0 \neq 0$	Theorem 4.4
	$O(nr^{\iota} TP^{m})$: DP if $\zeta_0 = 0$	Theorem 4.4
	$O\left(\frac{n^{2m+3} L^{m+1}}{\varepsilon^{m+1}}\right)$: FPTAS if $\zeta_0 = 0$ and $\iota = 1$	Theorem 4.10
$Pm, \iota \mid nr-a, NM, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$	*NP*-hard in the ordinary sense	Lee and Liman [14]
	$O(nr^{2\iota} \mathcal{TP}^{(s+1)\iota} TP^{m-\iota})$: DP if $\zeta_0 \neq 0$	Theorem 4.7
	$O(nr^{\iota} \mathcal{TP}^{s\iota} TP^{m-\iota})$: DP if $\zeta_0 = 0$	Theorem 4.7
	$O\left(\frac{n^{2(s-1)\iota+2m+3} L^{(s-1)\iota+m+1}}{\varepsilon^{(s-1)\iota+m+1}}\right)$: FPTAS if $\zeta_0 = 0$ and $\iota = 1$	Theorem 4.11
$Pm, \iota \mid r-a, M, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$	*NP*-hard in the ordinary sense if $\iota \geq 2$; open if $\iota = 1$	Lee and Yu [16]
	$O(n^{\iota+1} r^{2\iota} TP^{m+\iota})$: DP if $\zeta_0 \neq 0$	Theorem 5.3
	$O(n^{\iota+1} r^{\iota} TP^{m})$: DP if $\zeta_0 = 0$	Theorem 5.3
$Pm, \iota \mid r-a, NM, DE, PDR \mid E(\sum_{j=1}^{n} C_j)$	*NP*-hard in the ordinary sense if $\iota \geq 2$; open if $\iota = 1$	Lee and Yu [16]
	$O(n^{\iota+1} r^{2\iota} \mathcal{TP}^{(s+1)\iota} TP^{m-\iota})$: DP if $\zeta_0 \neq 0$	Theorem 5.4
	$O(n^{\iota+1} r^{2\iota} \mathcal{TP}^{s\iota} TP^{m-\iota})$: DP if $\zeta_0 = 0$	Theorem 5.4

Please cite this article as: Yin Y, et al. Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.07.006

- Extend our model to the case where the disruption starting time is random that follows a given probability distribution.
- Extend our model to different machine environments, e.g., the flow shop.

Acknowledgements

We thank the editor, an associate editor, and anonymous referees for their helpful comments on earlier versions of our paper. Yin was supported in part by the National Natural Science Foundation of China (Nos. 11561036 and 71301022) and the Personnel Training Fund of Kunming University of Science and Technology under grant number KKSY201407098; and Cheng was supported in part by The Hong Kong Polytechnic University under the Fung Yiu King – Wing Hang Bank Endowed Professorship in Business Administration.

Appendix A

Proof of Lemma 4.8. We prove the lemma by induction on j. It is clear that the lemma holds for $j=1$. As the induction hypothesis, we assume that the lemma holds for any $j=k-1$, i.e., for any eliminated state $(u_1,u_2,q_1,f)\in\mathcal{F}_{(k-1)}$, there exists a state $(\widetilde{u_1},\widetilde{u_2},\widetilde{q_1},\widetilde{f})$ such that $\widetilde{u_1}\le u_1$, $\widetilde{u_2}\le\delta^{k-1}u_2$, $\widetilde{q_1}\le\delta^{k-1}q_1$, and $\widetilde{f}\le\delta^{k-1}f$. We show that the lemma holds for $j=k$.

Consider an arbitrary state $(u_1,u_2,q_1,f)\in\mathcal{F}_{(k)}$. First, we assume that job J_j finishes processing no later than r on machine M_1 in the corresponding partial schedule. Then there must be $u_1\le r$ by the constraint. While implementing algorithm *NRMDPDAA*$_\varepsilon$, the state (u_1,u_2,q_1,f) is constructed from $((u_1-p_j)/(1+p_ja),u_2,q_1,f-u_1)\in\mathcal{F}_{(k-1)}$. According to the induction hypothesis, there exists a state $(\widehat{u_1},\widehat{u_2},\widehat{q_1},\widehat{f})\in\mathcal{F}_{(k-1)}$ such that $\widehat{u_1}\le(u_1-p_j)/(1+p_ja)$, $\widehat{u_2}\le\delta^{k-1}u_2$, $\widehat{q_1}\le\delta^{k-1}q_1$, and $\widehat{f}\le\delta^{k-1}(f-u_1)$. Since

$$\widehat{u_1}+p_j(1+a\widehat{u_1})=(\widehat{u_1}(1+p_ja)+p_j)\le u_1\le r,$$

the state $(\widehat{u_1}+p_j(1+a\widehat{u_1}),\widehat{u_2},\widehat{q_1},\widehat{f}+\widehat{u_1}+p_j(1+a\widehat{u_1}))$ is generated during the state generation process. However, this state may be eliminated by another state $(\widetilde{u_1},\widetilde{u_2},\widetilde{q_1},\widetilde{f})$ through the elimination rules in algorithm *NRMDPDAA*$_\varepsilon$. If it is eliminated by $(\widetilde{u_1},\widetilde{u_2},\widetilde{q_1},\widetilde{f})$ through elimination rule (1), we have

(a) $\widetilde{u_1}\le\widehat{u_1}+p_j(1+a\widehat{u_1})\le u_1$,
(b) $\widetilde{u_2}\le\delta\widehat{u_2}\le\delta^ku_2$,
(c) $\widetilde{q_1}\le\delta\widehat{q_1}\le\delta^kq_1$, and
(d) $\widetilde{f}\le\delta(\widehat{f}+\widehat{u_1}+p_j(1+a\widehat{u_1}))\le\delta(\delta^{k-1}(f-u_1)+u_1)\le\delta\delta^{k-1}(f-u_1+u_1)=\delta^kf$.

Otherwise, by the above proof, we have $\widetilde{u_1}=\widehat{u_1}+p_j(1+a\widehat{u_1})\le u_1$, $\widetilde{u_2}\le\widehat{u_2}\le\delta\widehat{u_2}\le\delta^ku_2$, $\widetilde{q_1}\le\widehat{q_1}\le\delta\widehat{q_1}\le\delta^kq_1$, and $\widetilde{f}\le\widehat{f}+\widehat{u_1}+p_j(1+a\widehat{u_1})\le\delta^{k-1}f\le\delta^kf$.

It follows that the induction hypothesis holds for $j=k$ when job J_j finishes processing no later than r on machine M_1 in the corresponding partial schedule.

Now, we turn to the case where job J_j finishes processing after r on machine M_1 in the corresponding partial schedule. While implementing algorithm *NRMDPDAA*$_\varepsilon$, the state (u_1,u_2,q_1,f) is constructed from $(u_1,u_2,(q_1-p_j)/(1+p_ja),\ f-\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+q_1))\in\mathcal{F}_{(k-1)}$. According to the induction hypothesis, there exists a state $(\widehat{u_1},\widehat{u_2},\widehat{q_1},\widehat{f})\in\mathcal{F}_{(k-1)}$ such that $\widehat{u_1}\le u_1$, $\widehat{u_2}\le\delta^{k-1}u_2$, $\widehat{q_1}\le\delta^{k-1}(q_1-p_j)/(1+p_ja)$, and $\widehat{f}\le\delta^{k-1}(f-\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+q_1))$. During the state generation process, the state $(\widehat{u_1},\widehat{u_2},\widehat{q_1}+p_j(1+a\widehat{q_1}),\widehat{f}+\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+\widehat{q_1}+p_j(1+a\widehat{q_1})))$ is generated. However, this state may be eliminated by another state $(\widetilde{u_1},\widetilde{u_2},\widetilde{q_1},\widetilde{f})$ through the elimination rules in algorithm *NRMDPDAA*$_\varepsilon$. If it is eliminated by $(\widetilde{u_1},\widetilde{u_2},\widetilde{q_1},\widetilde{f})$ through elimination rule (1), we have

(a) $\widetilde{u_1}\le\widehat{u_1}\le u_1$,
(b) $\widetilde{u_2}\le\delta\widehat{u_2}\le\delta^ku_2$,
(c) $\widetilde{q_1}\le\delta(\widehat{q_1}+p_j(1+a\widehat{q_1}))=\delta(p_j+\widehat{q_1}(1+p_ja))\le\delta(p_j+(1+p_ja)\delta^{k-1}(q_1-p_j)/(1+p_ja))$
$=\delta(p_j+\delta^{k-1}(q_1-p_j))\le\delta\delta^{k-1}(p_j+(q_1-p_j))=\delta^kq_1$,

and

$$\begin{aligned}\text{(d) }\widetilde{f}&\le\delta\left(\widehat{f}+\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+\widehat{q_1}+p_j(1+a\widehat{q_1}))\right)\\&\le\delta\left(\delta^{k-1}\left(f-\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+q_1)\right)\right.\\&\quad\left.+\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+\widehat{q_1}+p_j(1+a\widehat{q_1}))\right)\\&\le\delta\left(\delta^{k-1}\left(f-\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+q_1)\right)\right.\end{aligned}$$

Please cite this article as: Yin Y, et al. Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.07.006

$$+\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+\delta^{k-1}q_1)\Bigg)$$
$$\le\delta\Bigg(\delta^{k-1}\Bigg(f-\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+q_1)\Bigg)$$
$$+\delta^{k-1}\Bigg(\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+q_1)\Bigg)\Bigg)=\delta^k f.$$

Otherwise, by the above proof, we have $\widetilde{u_1}=\widehat{u_1}\le u_1$, $\widetilde{u_2}\le\widehat{u_2}\le\delta\widehat{u_2}\le\delta^k u_2$, $\widetilde{q_1}\le\widehat{q_1}+p_j(1+a\widehat{q_1})\le\delta(\widehat{q_1}+p_j(1+a\widehat{q_1}))\le\delta^k q_1$, and $\widetilde{f}\le\widehat{f}+\sum_{\gamma=1}^{s}\zeta_\gamma(r+\max\{\gamma,\kappa\}+\widehat{q_1}+p_j(1+a\widehat{q_1}))\le\delta^{k-1}f\le\delta^k f$.

It follows that the induction hypothesis holds for $j=k$ when job J_j finishes processing after r on machine M_1 in the corresponding partial schedule.

Finally, we consider the case where job J_j is scheduled on machine M_2 in the corresponding partial schedule. While implementing algorithm *NRMDPDAA*$_\varepsilon$, the state (u_1,u_2,q_1,f) is constructed from $(u_1,(u_2-p_j)/(1+p_ja),q_1,f-u_2)\in\mathcal{F}_{(k-1)}$. According to the induction hypothesis, there exists a state $(\widehat{u_1},\widehat{u_2},\widehat{q_1},\widehat{f})\in\mathcal{F}_{(k-1)}$ such that $\widehat{u_1}\le u_1$, $\widehat{u_2}\le\delta^{k-1}(u_2-p_j)/(1+p_ja)$, $\widehat{q_1}\le\delta^{k-1}q_1$, and $\widehat{f}\le\delta^{k-1}(f-u_2)$. During the state generation process, the state $(\widehat{u_1},\widehat{u_2}+p_j(1+a\widehat{u_2}),\widehat{q_1},\widehat{f}+\widehat{u_2}+p_j(1+a\widehat{u_2}))$ is generated. However, this state may be eliminated by another state $(\widetilde{u_1},\widetilde{u_2},\widetilde{q_1},\widetilde{f})$ through the elimination rules in algorithm *NRMDPDAA*$_\varepsilon$. If it is eliminated by $(\widetilde{u_1},\widetilde{u_2},\widetilde{q_1},\widetilde{f})$ through elimination rule (1), we have

(a) $\widetilde{u_1}\le\widehat{u_1}\le u_1$,

(b) $\widetilde{u_2}\le\delta(\widehat{u_2}+p_j(1+a\widehat{u_2}))=\delta(p_j+\widehat{u_2}(1+p_ja))\le\delta(p_j+(1+p_ja)\delta^{k-1}(qu_2-p_j)/(1+p_ja))$
$=\delta(p_j+\delta^{k-1}(u_2-p_j))\le\delta\delta^{k-1}(p_j+(u_2-p_j))=\delta^k u_2$,

(c) $\widetilde{q_1}\le\delta\widetilde{q_1}\le\delta\delta^{k-1}q_1=\delta^k q_1$, and

(d) $\widetilde{f}\le\delta(\widehat{f}+\widehat{u_2}+p_j(1+a\widehat{u_2}))$
$\le\delta(\delta^{k-1}(f-u_2)+\widehat{u_2}+p_j(1+a\widehat{u_2}))$
$\le\delta(\delta^{k-1}(f-u_2)+\delta^{k-1}u_2)=\delta^k f$.

Otherwise, by the above proof, we have $\widetilde{u_1}=\widehat{u_1}\le u_1$, $\widetilde{u_2}\le\widehat{u_2}+p_j(1+a\widehat{u_2})\le\delta(\widehat{u_2}+p_j(1+a\widehat{u_2}))\le\delta^k u_2$, $\widetilde{q_1}\le\widehat{q_1}\le\delta\widehat{q_1}\le\delta^k q_1$, and $\widetilde{f}\le\widehat{f}+\widehat{u_2}+p_j(1+a\widehat{u_2})\le\delta^{k-1}f\le\delta^k f$.

It follows that the induction hypothesis holds for $j=k$ when job J_j is scheduled on machine M_2 in the corresponding partial schedule. Thus, the induction hypothesis holds in each case and the result follows. □

References

[1] Bean JC, Birge JR, Mittenthal J, Noon CE. Matchup scheduling with multiple resources, release dates and disruptions. Operations Research 1991;39:470–83.
[2] Birge J, Glazebrook KD. Assessing the effects of machine breakdowns in stochastic scheduling. Operations Research Letters 1988;7:267–71.
[3] Camci F. Maintenance scheduling of geographically distributed assets with prognostics information. European Journal of Operational Research 2015;245:506–16.
[4] Chen JS. Optimization models for the machine scheduling problem with a single flexible maintenance activity. Engineering Optimization 2006;38:53–71.
[5] Finke G, Gara-Ali A, Espinouse M-L, Jost V, Moncel J. Unified matrix approach to solve production-maintenance problems on a single machine. Omega, http://dx.doi.org/10.1016/j.omega.2016.02.005.
[6] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic machine scheduling: a survey. Annals of Discrete Mathematics 1979;5:287–326.
[7] Hall NG, Potts CN. Rescheduling for job unavailability. Operations Research 2010;58:746–55.
[8] Ji M, He Y, Cheng TCE. Scheduling linear deteriorating jobs with an availability constraint on a single machine. Theoretical Computer Science 2006;362:115–26.
[9] Kacem I. Approximation algorithm for the weighted flow-time minimization on a single machine with a fixed non-availability interval. Computers & Industrial Engineering 2008;54:401–10.
[10] Kacem I. Approximation algorithms for makespan minimization with positive tails on a single machine with a fixed nonavailability interval. Journal of Combinatorial Optimization 2009;17:117–33.
[11] Kacem I, Levner E. An improved approximation scheme for scheduling a maintenance and proportional deteriorating jobs. Journal of Industrial and Management Optimization 2016;1:811–7.
[12] Lee C-Y. Machine scheduling with an availability constraint. Journal of Global Optimization 1996;9:363–84.
[13] Lee C-Y, Lei L, Pinedo M. Current trends in deterministic scheduling. Annals of Operations Research 1997;70:1–41.
[14] Lee C-Y, Liman SD. Single machine flow-time scheduling with scheduled maintenance. Acta Informatica 1992;29:375–82.
[15] Lee C-Y, Yu G. Single machine scheduling under potential disruption. Operations Research Letters 2007;35:541–8.
[16] Lee C-Y, Yu G. Parallel-machine scheduling under potential disruption. Optimization Letters 2008;2:27–37.
[17] Levin A, Mosheiov G, Sarig A. Scheduling a maintenance activity on parallel identical machines. Naval Research Logistics 2009;56:33–41.
[18] Luo W, Liu F. On single-machine scheduling with workload-dependent maintenance duration. Omega, http://dx.doi.org/10.1016/j.omega.2016.06.008.
[19] Ma Y, Chu C, Zuo C. A survey of scheduling with deterministic machine availability constraints. Computers & Industrial Engineering 2010;58:199–211.
[20] Mor B, Mosheiov G. Batch scheduling with a rate-modifying maintenance activity to minimize total flowtime. International Journal of Production Economics 2014;153:238–42.
[21] Rustogi K, Strusevich VA. Single machine scheduling with general positional deterioration and rate-modifying maintenance. Omega 2012;40:791–804.
[22] Rustogi K, Strusevich VA. Combining time and position dependent effects on a single machine subject to rate-modifying activities. Omega 2014;42:166–78.

Please cite this article as: Yin Y, et al. Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.07.006

[23] Sanlaville E, Schmidt G. Machine scheduling with availability constraints. Acta Informatica 1998;35:795–811.
[24] Sahni SK. Algorithms for scheduling independent tasks. Journal of the ACM 1976;23(1):116–27.
[25] Schmidt G. Scheduling with limited machine availability. European Journal of Operational Research 2000;121:1–15.
[26] Valdez-Flores C, Feldman RM. A survey of preventive maintenance models for stochastically deteriorating single-unit systems. Naval Research Logistics 1989;36:419–46.
[27] Wang JB, Cheng TCE. Scheduling problems with the effects of deterioration and learning. Asia-Pacific Journal of Operational Research 2007;24(2):245–61.
[28] Xu D, Wan L, Liu A, Yang DL. Single machine total completion time scheduling problem with workload-dependent maintenance duration. Omega 2015;52:101–6.
[29] Yang DL, Yang SJ. Unrelated parallel-machine scheduling problems with multiple rate-modifying activities. Information Sciences 2013;235:280–6.
[30] Yang SJ, Yang DL, Cheng TCE. Single-machine due-window assignment and scheduling with job-dependent aging effects and deteriorating maintenance. Computers and Operations Research 2010;37:1510–4.
[31] Yin Y, Wang DJ, Cheng TCE. Rescheduling on identical parallel machines with machine disruptions to minimize total completion time. European Journal of Operational Research 2016;252:737–49.
[32] Yin Y, Wu W-H, Cheng TCE, Wu C-C. Due date assignment and single-machine scheduling with generalized positional deteriorating jobs and deteriorating multi-maintenance activities. International Journal of Production Research 2014;52:2311–26.
[33] Yin Y, Xu J, Cheng TCE, Wu C-C, Wang DJ. Approximation schemes for single-machine scheduling with a fixed maintenance activity to minimize the total amount of late work. Naval Research Logistics 2016;63:172–83.
[34] Yin Y, Ye D, Zhang G. Single machine batch scheduling to minimize the sum of total flow time and batch delivery cost with an unavailability interval. Information Sciences 2014;274:310–22.

Please cite this article as: Yin Y, et al. Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. Omega (2016), http://dx.doi.org/10.1016/j.omega.2016.07.006