ELSEVIER

Discrete Applied Mathematics 117 (2002) 195–209

DISCRETE APPLIED MATHEMATICS

Complexity analysis of job-shop scheduling with deteriorating jobs

Gur Mosheiov *

School of Business Administration and Department of Statistics, The Hebrew University of Jerusalem, Mount Scopus, 91905 Jerusalem, Israel

Received 3 August 1998; revised 2 November 2000; accepted 6 November 2000

Abstract

This paper addresses job-shop scheduling problems with deteriorating jobs, i.e. jobs whose processing times are an increasing function of their starting time. A simple linear deterioration is assumed and our objective is makespan minimization. We provide a complete analysis of the complexity of flow-shops, open-shops and job-shop problems. We introduce a polynomial-time algorithm for the two-machine flow-shop, and prove NP-hardness when an arbitrary number of machines (three and above) is assumed. Similarly, we introduce a polynomial-time algorithm for the two-machine open-shop, and prove NP-hardness when an arbitrary number of machines (three and above) is assumed. Finally, we prove NP-hardness of the job-shop problem even for two machines. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Deterministic scheduling; Flow-shop; Open-shop; Job-shop; Deteriorating jobs

1. Introduction

In recent years, we have become aware of a growing interest in studying scheduling problems of *deteriorating jobs*, i.e. jobs whose processing time is an increasing function of their starting time. Such deterioration appears e.g. in scheduling maintenance jobs, where any delay in processing a job is penalized and often implies additional time for accomplishing the job; see [9,12] for a list of applications. In their recent comprehensive survey, Cheng and Ding [3] refer to about 30 papers dealing with different combinations of machine settings and deterioration types. Most of these studies focus on single machine settings; see e.g. [1,11,15]. Chen [2] and Mosheiov [14] studied scheduling deteriorating jobs in a *multi-machine* setting. They assumed linear deterioration and parallel identical machines. Chen considered minimum flow time and Mosheiov studied

* Tel.: +972-2-588-3235; fax: +972-2-588-1341.
E-mail address: msomer@mscc.huji.ac.il (G. Mosheiov).

0166-218X/02/$ - see front matter
PII: S0166-218X(00)00385-1

makespan minimization. Mosheiov [13] also considered makespan minimization on a general flow-shop. All these (multi-machine) scheduling problems were shown to be NP-hard and the above papers focus on the introduction of efficient heuristics.

In this paper we study makespan minimization of deteriorating jobs in *shop* settings, i.e. when each job consists of a number of operations which need to be processed on different machines. We study all classical shop environments: *flow-shops* (when operations of all jobs need to be processed in the same order, i.e. jobs must follow the same route), *open-shops* (when job-routes are immaterial), and *job-shops* (when job-routes are job-dependent). In most studies on scheduling deteriorating jobs, the deterioration type is linear (i.e. a general affine deterioration function of the job starting time is assumed); some papers consider piece-wise linear and others assume step functions; see [3]. We consider *simple linear deterioration* (see [2,12]), i.e. we assume that the job processing time P_j is given as a simple function $P_j = \alpha_j t$, where t is the job starting time and α_j is the fixed job-dependent deterioration rate. [Clearly, all the NP-hardness proofs introduced in this paper for the case of simple linear deterioration, also hold for general affine deterioration functions.] All the models considered in this paper assume deterioration rates which are both job-dependent and machine-dependent.

This paper focuses on complexity results. We provide a complete analysis of the complexity of job-shop scheduling problems with deteriorating jobs. We introduce a polynomial-time algorithm for the two-machine flow-shop, and prove NP-hardness when an arbitrary number of machines (three and above) is assumed. Similarly, we introduce a polynomial-time algorithm for the two-machine open-shop, and prove NP-hardness when an arbitrary number of machines (three and above) is assumed. Finally, we prove NP-hardness of the job-shop problem even for two machines. All NP-hardness proofs are based on reductions from the Subset Product problem which has been shown to be NP-hard in the ordinary sense; see [4,8].

It is worth noting that all our complexity results are consistent with the complexities of the classical versions of the problems (with no job deterioration), i.e the classical two-machine flow-shop and open-shop have polynomial-time solutions, whereas the three-machine version in both cases and the classical two-machine job-shop are known to be hard. This consistency however, is not always warranted as reflected by Chen's proof [2] that minimizing flow-time on parallel identical machines with simple linear deterioration is hard (whereas the classical version of this problem is polynomially solved).

In Sections 2, 3 and 4 we study the flow-shop problem, the open-shop problem and the job-shop problem, respectively.

2. Flow-shops

N jobs are to be processed on an M-machine flow-shop. P_{ij} denotes the processing time of job i ($i = 1,\ldots,N$) on machine j ($j = 1,\ldots,M$). P_{ij} is a simple linear function of the job's starting time, i.e. $P_{ij} = \alpha_{ij} t$, where $\alpha_{ij} \geqslant 0$ is the deterioration rate of job i

on machine j, and $t \geqslant t_0$ is the job starting-time. All jobs are available for processing at time $t_0 > 0$. The objective is to minimize makespan. We assume unlimited intermediate storage between successive machines (see [13] for some discussion on the case of limited intermediate storage). We also restrict ourselves to permutation schedules only.

Let $C_{ij}(q)$ denote the completion time of job i on machine j under some schedule q. Let $C_{(i)j}(q)$ denote the completion time of the ith job on machine j under schedule q. Thus, makespan is denoted by $C_{\max}(q) = C_{(N)M}(q)$. Since unlimited intermediate storage is assumed, clearly an optimal schedule exists with no idle times between consecutive jobs on machine 1. Therefore, the completion time of the first job on machine 1 is given by $C_{(1)1}(q) \equiv C_{(1)1} = t_0 + \alpha_{(1)1} t_0 = t_0(1 + \alpha_{(1)1})$. The completion time of the ith job on machine 1 is given by

$$C_{(i)1} = t_0 \prod_{j=1}^{i} (1 + \alpha_{(j)1}), \quad i = 1, \ldots, N. \tag{1}$$

Note that t_0 is a common factor in all completion times, and does not affect the optimal schedule. For convenience let $t_0 = 1$, if not specified otherwise.

We focus first on the makespan minimization problem on a *two-machine* flow-shop. We show that similar to the classical two-machine flow-shop (i.e. when processing times are fixed over time), a polynomial-time algorithm exists when linear deterioration is assumed. For ease of exposition we denote α_{i1} by α_i, and α_{i2} by β_i, $i = 1, \ldots, N$.

Lemma 1. *There exists an optimal schedule in which the job sequence is identical on both machines.*

Proof. Assume that an optimal schedule q exists with job k directly preceding job l on machine 1 but job l preceding job k on machine 2. Let $C_{k1}(q)=a, C_{l1}(q)=C_{k1}(1+\alpha_l)=b$ and $C_{p2}(q) = c$, where p is the last job processed prior to l on machine 2. Then, the starting time of job l on machine 2 is given by $d = \max\{b, c\}$.

Consider schedule q_1 obtained from q by interchanging jobs k and l on machine 1. Note that $C_{k1}(q_1) = C_{l1}(q) = b$, by (1). Clearly $C_{l1}(q_1) < C_{l1}(q)$. Thus, under q_1 the starting time of job l on machine 2 is given by $\max\{C_{l1}(q_1), c\} \leqslant \max\{b, c\} = d$. Since the sequence on machine 2 is not changed, this interchange does not increase the completion time of any job. We conclude that schedule q_1 is at least as good as schedule q. This procedure of interchanging jobs, may be repeated until an optimal schedule with the same job-sequence on both machines is obtained. □

The conclusion of Lemma 1 is that only permutation schedules should be considered for this problem. The classical two machine flow-shop is solved by the well-known $O(N \log N)$ Johnson Algorithm [7]. Johnson Algorithm consists of first scheduling the set of jobs with $P_{i1} \leqslant P_{i2}$ in non-decreasing order of P_{i1}, and then scheduling the set of jobs with $P_{i1} > P_{i2}$ in non-increasing order of P_{i2}. When simple linear deterioration

is assumed, the optimal schedule is obtained by modifying this procedure as follows: we first schedule the set of jobs with $\alpha_i \leqslant \beta_i$ (Set 1) in a non-decreasing order of α_i, and then we schedule the remaining set (Set 2 contains jobs with $\alpha_i > \beta_i$) in a non-increasing order of β_i. Clearly, the running-time remains $O(N \log N)$. A formal procedure is given below:

Algorithm SOLVE_FS2 for solving the two-machine flow-shop:

Step 0: Let $k = 1$; $l = n$;
$J = \{1, 2, \ldots, n\}$ = the current set of unscheduled jobs.

Step 1: Find $\lambda = \min\{\alpha_i, \beta_i : j \in J\}$, i.e. find the job with the smallest deterioration rate on any of the machines, in the set of the unscheduled jobs.

Step 2: If $\lambda = \alpha_p$ for some p (i.e. the smallest deterioration rate is on machine 1):
Schedule job p in the kth position,
$J = J \setminus \{p\}$,
$k = k + 1$.
If $J \neq \emptyset$ go to Step 2. Otherwise stop.
Else ($\lambda = \beta_q$ for some q, i.e. the smallest deterioration rate is on machine 2):
Schedule job q in the lth position,
$J = J \setminus \{q\}$,
$l = l - 1$.
If $J \neq \emptyset$ go to Step 2. Otherwise stop.

Theorem 1. *Algorithm SOLVE_FS2 produces an optimal schedule for the two-machine flow-shop.*

Proof. Let q denote an optimal schedule which is not obtained by Algorithm SOLVE_FS2. Schedule q must contain two adjacent jobs, say j followed by k, such that one of the following three conditions is satisfied:

Condition 1. $\alpha_j > \beta_j$ and $\alpha_k \leqslant \beta_k$ (job j is in Set 2 and job k is in Set 1).

Condition 2. $\alpha_j \leqslant \beta_j$, $\alpha_k \leqslant \beta_k$ and $\alpha_j > \alpha_k$ (jobs j and k are in Set 1 and arranged in a decreasing order of α).

Condition 3. $\alpha_j > \beta_j$, $\alpha_k > \beta_k$ and $\beta_j < \beta_k$ (jobs j and k are in Set 2 and arranged in an increasing order of β).

Denote by q_1 the schedule obtained after a pairwise interchange of jobs j and k. It suffices to show that under any of these conditions the makespan can only be reduced.

Recall that there are no idle times on machine 1 after time t_0. Hence $C_k^{(1)}(q) = C_j^{(1)}(q_1)$. Thus, it remains to check the effect of this interchange on the maximal completion time (of the two jobs) on machine 2. Specifically, we have to prove that $C_j^{(2)}(q_1) \leqslant C_k^{(2)}(q)$.

Let

a = the last completion time prior to jobs j and k on machine 1 in q,

d = the last completion time prior to jobs j and k on machine 2 in q.

Clearly, a and d remain unchanged in schedule q_1.

The completion time of job k on machine 2 under schedule q is

$$C_k^{(2)} = [\max\{\max\{d, a(1+\alpha_j)\}(1+\beta_j), a(1+\alpha_j)(1+\alpha_k)\}](1+\beta_k)$$
$$= \max\{d(1+\beta_j)(1+\beta_k), a(1+\alpha_j)(1+\beta_j)(1+\beta_k),$$
$$a(1+\alpha_j)(1+\alpha_k)(1+\beta_k)\}. \quad (2)$$

The completion time of job j on machine 2 under schedule q_1 is

$$\max\{d(1+\beta_k)(1+\beta_j), a(1+\alpha_k)(1+\beta_k)(1+\beta_j), a(1+\alpha_k)(1+\alpha_j)(1+\beta_j)\}. \quad (3)$$

It is easily verified that under the first condition:
first term in (3) = first term in (2),
second term in (3) $\leqslant$ third term in (2),
third term in (3) $\leqslant$ second term in (2).
Under the second condition:
first term in (3) = first term in (2),
second term in (3) $\leqslant$ second term in (2),
third term in (3) $\leqslant$ second term in (2).
Under the third condition:
first term in (3) = first term in (2),
second term in (3) $\leqslant$ third term in (2),
third term in (3) $\leqslant$ third term in (2).
We conclude that in all cases $C_j^{(2)}(q_1) \leqslant C_k^{(2)}(q)$. By repeating this procedure we obtain an optimal schedule without any of these conditions, i.e. a schedule produced by Algorithm SOLVE_FS2. □

Next, we prove that the general *multi-machine* case (three machines or more) is NP-hard, as in classical scheduling. The associated decision problem of makespan minimization on an M-machine flow-shop with simple linear deterioration is the following:

FS: Given a set $J = \{1, 2, \ldots, N\}$ of jobs, an M-machine flow-shop ($M \in Z^+$), a deterioration rate $\alpha_{ij} \in R^+ \cup \{0\}$ of job i on machine j ($i = 1, \ldots, N, j = 1, \ldots, M$), a common release date t_0, and a positive integer D, is there a schedule q such that $C_{\max}(q) \leqslant D$?

We transform the "Subset Product" (which was shown to be NP-complete in the ordinary sense, see [4] and a correction of the complexity status in [8]), into problem **FS**. The Subset Product Problem (**SP**) can be stated as follows:

SP: Given a set $A = \{1, 2, \ldots, v\}$ and a size $a_i \in Z^+$ for each $i \in A$, is there a subset $A' \subseteq A$, such that $\prod_{i \in A'} a_i = \prod_{i \in A \setminus A'} a_i = \sqrt{\prod_{i \in A} a_i} \equiv B$?

Theorem 2. *Problem* **FS** *is* NP-*complete even for* 3 *machines.*

Proof. Clearly **FS** $\in$ NP. Let I_{SP} be an instance of **SP**. In order to show that **SP** can be reduced to **FS**, we construct the corresponding instance I_{FS} of **FS**. Let $N = v + 3$ and $M = 3$.

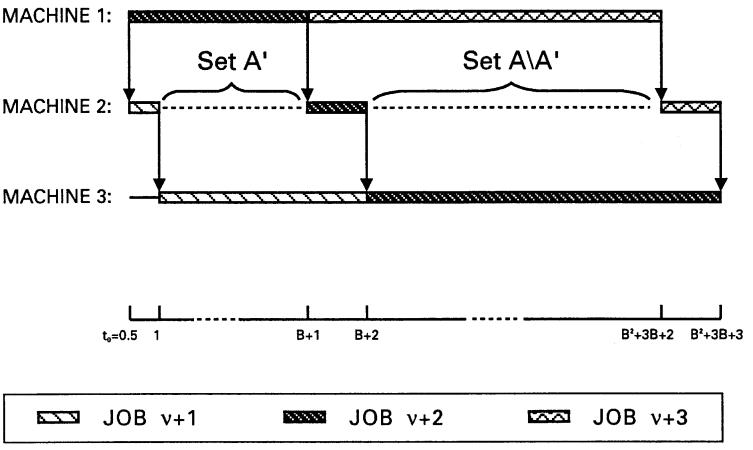

Fig. 1. A three-machine flow-shop.

For convenience we denote $\alpha_{i1} = \alpha_i$, $\alpha_{i2} = \beta_i$, $\alpha_{i3} = \gamma_i$:

$$\alpha_i = 0, \quad \beta_i = a_i - 1, \quad \gamma_i = 0, \quad i = 1, \ldots, v,$$

$$\alpha_{v+1} = 0, \quad \beta_{v+1} = 1, \quad \gamma_{v+1} = B + 1,$$

$$\alpha_{v+2} = 2B + 1, \quad \beta_{v+2} = \frac{1}{B+1}, \quad \gamma_{v+2} = \frac{B^2 + 2B + 1}{B+2},$$

$$\alpha_{v+3} = \frac{B^2 + 2B + 1}{B+1}, \quad \beta_{v+3} = \frac{1}{B^2 + 3B + 2}, \quad \gamma_{v+3} = 0,$$

$$t_0 = 1/2 \quad \text{and} \quad D = B^2 + 3B + 3.$$

Obviously, this construction can be accomplished in polynomial time.

We show that the following two statements hold:

(i) If for instance I_{SP}, there exists a subset A' such that $\prod_{i\in A'} a_i = \prod_{i\in A\setminus A'} a_i = \sqrt{\prod_{i\in A} a_i}$, then for instance I_P there exists a schedule q such that $C_{\max}(q) \leqslant D$.

For the given solution of I_{SP}, create a schedule q' as follows: Schedule jobs $v+1, v+2, v+3$ in this order. Note that the makespan is $B^2 + 3B + 3$ and that two gaps are left on machine 2: the first is at time 1 of length B, and the second is at time $B + 2$ of length $B^2 + 2B$; see Fig. 1.

Schedule the jobs with indices $i \in A'$ to fill the first gap (i.e. after job $v + 1$) and the remaining jobs to fill the second gap (after job $v + 2$). (The total load of the first subset of jobs is $t\prod_{i\in A'}(1+\beta_i) = 1\prod_{i\in A'} a_i = B$. The total load of the second subset of jobs is $t\prod_{i\in A\setminus A'}(1+\beta_i) = (B+2)\prod_{i\in A\setminus A'} a_i = B^2 + 2B$.)

(ii) If for I_{FS} there exists a schedule q with $C_{\max}(q) \leqslant D = B^2 + 3B + 3$, then for I_{SP} there exists a subset $A' \subseteq A$, such that $\prod_{i\in A'} a_i = \prod_{i\in A\setminus A'} a_i = \sqrt{\prod_{i\in A} a_i} = B$.

It is easily verified that if such q exists, then it must contain jobs $v+1, v+2, v+3$ in this order. (Any other sequence of these three jobs has $C_{\max} > D$.) As mentioned above, this schedule leaves two gaps on machine 2 (see Fig. 1). Consequently, the set of jobs $1,2,\ldots,v$ (say set A) must be scheduled in these gaps. It follows that a subset $A' \subseteq A$ exists such that $1\prod_{i\in A'}(1+\beta_i)\leqslant B$, and $(B+2)\prod_{i\in A\setminus A'}(1+\beta_i)\leqslant B^2+2B$, i.e. $\prod_{i\in A'}(1+\beta_i)\leqslant B$ and $\prod_{i\in A\setminus A'}(1+\beta_i)\leqslant B$. Since $\prod_{i\in A}(1+\beta_i)=\prod_{i\in A} a_i = B^2$, we obtain that $\prod_{i\in A'}(1+\beta_i)=\prod_{i\in A'} a_i = B$, and $\prod_{i\in A\setminus A'}(1+\beta_i)=\prod_{i\in A\setminus A'} a_i = B$. □

3. Open-shops

In practice, the route of the jobs is often immaterial and up to the scheduler to decide. As mentioned, when routes of the jobs are open, the model is referred to as an *open-shop*. In this section we prove that under simple linear deterioration, makespan minimization on a two-machine open-shop is polynomially solvable, and the general M-machine case is NP-hard. We focus first on the two-machine case.

Lemma 2. *A lower bound on the optimal makespan for the two-machine open-shop is*

$$\mathrm{LB} = MAX\left\{\prod_{j=1}^{N}(1+\alpha_j), \prod_{j=1}^{N}(1+\beta_j), \underset{1\leqslant j\leqslant N}{MAX}\{(1+\alpha_j)(1+\beta_j)\}\right\}. \tag{4}$$

Proof. The first component represents the total load on machine 1 with no idle time. The second component represents the total load on machine 2 with no idle time. The third component is the total processing of the largest job (on both machines) with no idle time between its two operations. Clearly, the makespan is at least as large as each one of these components. □

Gonzalez and Sahni [6] introduced an O(N) algorithm for the classical two-machine open-shop. In the following, we introduce a similar (O(N)) algorithm for solving the two-machine open-shop with linear deterioration. The algorithm always produces makespan which is identical with the above lower bound, and hence is optimal. For a given job j, we denote the operation processed on machine 1 by j^1, and the operation processed on machine 2 by j^2. For a given set of jobs S, we denote the set of operations processed on machine 1 by S^1, and the set of operations processed on machine 2 by S^2. Similarly, N^1 and N^2 denote the entire sets of operations processed on machines 1 and 2, respectively.

Algorithm SOLVE_OS2 for solving the two-machine open-shop:

Step 1: If for some job k $(k=1,\ldots,N)$

$$1+\alpha_k \geqslant \prod_{j\neq k}(1+\alpha_j) \quad \text{and} \quad 1+\beta_k \geqslant \prod_{j\neq k}(1+\beta_j): \tag{5}$$

[*Comment*: Both operations of job k are larger than all other operations on their machines, and the makespan is thus determined by job k.]

Schedule operation k^1 first on machine 1 and then (with no idle time) operation k^2 on machine 2. Schedule the remaining operations on machine 1 ($N^1 \backslash k^1$) at time t_0. Schedule the remaining operations on machine 2 ($N^2 \backslash k^2$) immediately after the completion time of k^1. See Fig. 2a. Stop.

Else (i.e. if (5) does not hold):

[*Comment*: A job which is larger than all other jobs in both operations does not exist.]

Step 2: Define

$$A = \{j \mid \alpha_j \geqslant \beta_j\}.$$

$$B = \{j \mid \alpha_j < \beta_j\}.$$

$$\beta_R = \underset{\{j \in A\}}{MAX}\{\beta_j\} \quad \text{(i.e., operation } R^2 \text{ has the largest } \beta \text{ value in the set } A^2\text{)}.$$

$$\alpha_L = \underset{\{j \in B\}}{MAX}\{\alpha_j\} \quad \text{(i.e., operation } L^1 \text{ has the largest } \alpha \text{ value in the set } B^1\text{)}.$$

Let $\bar{A} = A \setminus \{R\}$ and $\bar{B} = B \setminus \{L\}$.

Step 2a: Schedule operation L^1 first on machine 1, and then (with no idle time) operation L^2 on machine 2. Schedule the set $\bar{B}^1$ on machine 1 in any order after L^1, and the set $\bar{B}^2$ on machine 2 (in the same order) after L^2. This schedule is feasible since $\alpha_j < \beta_j$ for all $j \in \bar{B}$ (thus, every job in $\bar{B}$ on machine 1 is completed prior to the completion time of the previous job on machine 2).

Step 2b: Schedule operation R^2 last (and sufficiently late) on machine 2. Schedule R^1 (on machine 1) such that its completion time is identical to the starting time of R^2. Schedule the set $\bar{A}^1$ on machine 1 in any order immediately prior to R^1, and schedule the set $\bar{A}^2$ on machine 2 (in the same order) immediately prior to R^2. This schedule is feasible since $\alpha_j \geqslant \beta_j$ for all $j \in \bar{A}$ (thus, every job in $\bar{A}$ on machine 1 is completed prior to the completion time of the previous job on machine 2). See Fig. 2b.

[*Comment*: Note that it is always feasible to delay the set $\{L^2\} \cup \bar{B}^2$ on machine 2 (since $\alpha_j \leqslant \beta_j$ for all the jobs in this set), and to start the set $\bar{A}^1 \cup \{R^1\}$ earlier on machine 1 (since $\alpha_j > \beta_j$ for all the jobs in this set).]

$$\text{If } \prod_{\{j \neq L\}} (1 + \alpha_j) \geqslant \prod_{\{j \neq R\}} (1 + \beta_j): \tag{6}$$

[*Comment*: In this case the idle time on machine 1 (between $\bar{B}^1$ and $\bar{A}^1$) is less than or equal to the idle time on machine 2 (between $\bar{B}^2$ and $\bar{A}^2$).]

Step 3: Shift all the jobs in A (on both machines) to the left until there is no idle time on machine 1 (and there may be idle time on machine 2 between the sets $\bar{B}^2$ and $\bar{A}^2$). See Fig. 2c. Shift the set $\{L^2\} \cup \bar{B}^2$ on machine 2 to the

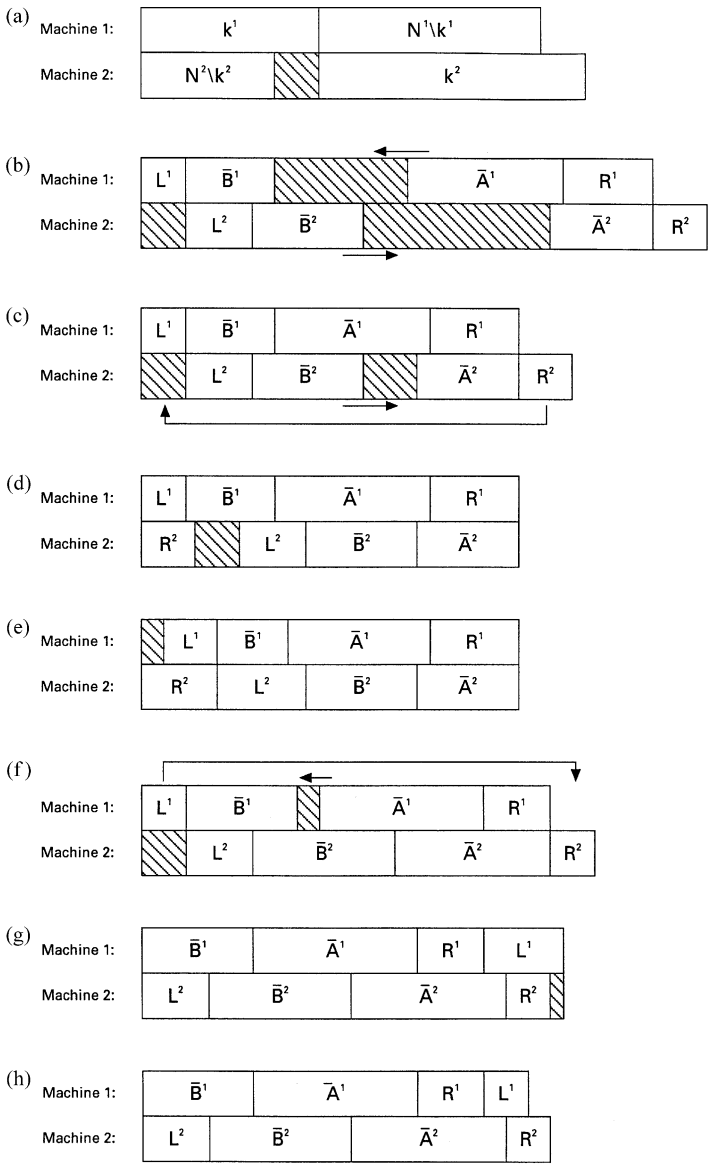

Fig. 2. A two-machine open-shop.

right (a feasible shift due to the above comment) to create idle time T_1 prior to the first operation on machine 2. ($T_1 = (\prod_{j=1}^{N}(1+\alpha_j))/(\prod_{\{j\neq R\}}(1+\beta_j))$).

If $1+\beta_R \leqslant T_1$: (7)

[*Comment*: In this case operation R^2 is not larger than the idle time T_1 prior to the first operation on machine 2, and R^2 can be shifted into this idle time.]

Step 3a: Move operation R^2 from the last position on machine 2 to the first position on machine 2. See Fig. 2d. Stop.
Else (if (7) does not hold, i.e. $1+\beta_R > T_1$):
[*Comment*: In this case operation R^2 is larger than the idle time T_1 prior to the first operation on machine 2. All operations on machine 2 must be shifted to the right in order to increase this idle time so that R^2 can be shifted into it.]

Step 3b: Delay all jobs on both machines equally until $1+\beta_R = T_1$, and schedule operation R^2 first on machine 2. See Fig. 2e. Stop.
Else (if (6) does not hold, i.e. $\prod_{\{j\neq L\}}(1+\alpha_j) < \prod_{\{j\neq R\}}(1+\beta_j)$):
[*Comment*: In this case the idle time on machine 1 (between $\bar{B}^1$ and $\bar{A}^1$) is larger than the idle time on machine 2 (between $\bar{B}^2$ and $\bar{A}^2$).]

Step 4: Shift all the jobs in A (on both machines) to the left until there is no idle time on machine 2 (and there may be idle time on machine 1 between the sets $\bar{B}^1$ and $\bar{A}^1$). See Fig. 2f. Shift the set $\bar{A}^1 \cup \{R^1\}$ on both machines 1 to the left (a feasible shift due to the above comment) to create idle time T_2 after the last operation on machine 1. $(T_2 = (\prod_{j=1}^{N}(1+\beta_j))/(\prod_{\{j\neq L\}}(1+\alpha_j)))$.

If $1+\alpha_L \leqslant T_2$: (8)

[*Comment*: In this case operation L^1 is not larger than the idle time T_2 after the last operation on machine 1, and L^1 can be moved into this idle time without affecting the makespan which is obtained on machine 2.]

Step 4a: Move operation L^1 from the first position on machine 1 to the last position on machine 1. An identical idle-time is created prior to the first operation on both machines. Shift all jobs on both machines to the left, to start at time t_0. See Fig. 2g. Stop. Else (if (8) does not hold, i.e. $1+\alpha_L > T_2$):
[*Comment*: In this case operation L^1 is larger than the idle time T_2 after the last operation on machine 1. When L_1 is shifted to the last position on machine 1, it is completed after the latest completion time on machine 2 and the makespan is obtained on machine 1.]

Step 4b: Move operation L^1 from the first position on machine 1 to the last position on machine 1. An identical idle-time is created prior to the first operation on both machine. Shift all jobs on both machines to the left, to start at time t_0. See Fig. 2h. Stop.

Theorem 3. *Algorithm SOLVE_OS2 produces an optimal schedule for the two-machine open-shop.*

Proof. It is easily verified that SOLVE_OS2 produces a feasible schedule, i.e. that no two operations of a given job overlap. There are five different schedules that may result from SOLVE_OS2, specified in steps 1, 3a, 3b, 4a and 4b. The schedule obtained in Step 1 leads to $C_{\max} = MAX_{1\leqslant j\leqslant N}\{(1+\alpha_j)(1+\beta_j)\}$ which is equal to the lower bound (4); see Fig. 2a. The schedule obtained in Step 3a leads to $C_{\max} = \prod_{j=1}^{N}(1+\alpha_j)$,

and the schedule obtained in Step 3b has $C_{\max} = \prod_{j=1}^{N}(1+\beta_j)$; see Figs. 2d and 2e, respectively. In both cases, the makespan is equal to lower bound (4). The schedule obtained in Step 4a results with $C_{\max} = \prod_{j=1}^{N}(1+\alpha_j)$, and the schedule obtained in Step 4b has $C_{\max} = \prod_{j=1}^{N}(1+\beta_j)$; see Figs. 2g and 2h, respectively. Again, the resulting makespan is equal to the lower bound. We conclude that SOLVE_OS2 produces an optimal schedule because in all cases the makespan value is identical to the lower bound. □

We now prove that the general M-machine open-shop is NP-hard. The associated decision problem of makespan minimization on an M-machine open-shop with simple linear deterioration is the following:

OS: Given a set $J = \{1,2,\ldots,N\}$ of jobs, an M-machine open-shop ($M \in Z^+$), a deterioration rate $\alpha_{ij} \in R^+ \cup \{0\}$ of job i on machine j ($i = 1,\ldots,N, j = 1,\ldots,M$), a common release date t_0, and a positive integer D, is there a schedule q such that $C_{\max}(q) \leqslant D$?

As in the case of makespan minimization on the M-machine flow-hop, we use "Subset Product" for the reduction.

Theorem 4. *Problem* **OS** *is* NP-*complete even for* 3 *machines.*

Proof. Clearly **OS** $\in$ NP. Let I_{SP} be an instance of **SP**. We construct the corresponding instance I_{OS} of **OS**. Let $N = v+1$ and $M = 3$.

Again we denote $\alpha_{i1} = \alpha_i,\ \alpha_{i2} = \beta_i,\ \alpha_{i3} = \gamma_i$:

$$\alpha_i = a_i - 1, \quad \beta_i = a_i - 1, \quad \gamma_i = a_i - 1, \quad i = 1,\ldots,v,$$

$$\alpha_{v+1} = \beta_{v+1} = \gamma_{v+1} = B - 1,$$

$$t_0 = 1 \quad \text{and} \quad D = B^3.$$

This construction can be accomplished in polynomial time.

We show that the following two statements hold:

(i) If for instance I_{SP}, there exists a subset A' such that $\prod_{i\in A'} a_i = \prod_{i\in A\setminus A'} a_i = \sqrt{\prod_{i\in A} a_i}$, then for instance I_{OS} there exists a schedule q such that $C_{\max}(q) \leqslant D$.

For the given solution of I_{SP}, create a schedule q' as follows: Schedule the three operations of job $v+1$ on machines 1, 2 and 3 in this order. (Start operation 1 on machine 1 at time $t = 1$.) The completion time on machine 3 is B^3. Schedule the jobs with indices $i \in A'$ on machine 2 starting at time 1. Schedule the remaining jobs at the same time interval on machine 3. Then schedule these two sets on machines 3 and 1 respectively, starting at time B. Finally, schedule these two sets on machines 1 and 2, respectively, starting at time B^2. The resulting schedule has $C_{\max}(q') = D = B^3$; see Fig. 3.

(ii) If there is no solution for **SP**, then there is no solution for **OS** (i.e. there is no schedule q such that $C_{\max}(q) \leqslant D$). Any optimal schedule must have no idle time between the operations of job $v+1$ (otherwise, the completion of the last operation of this

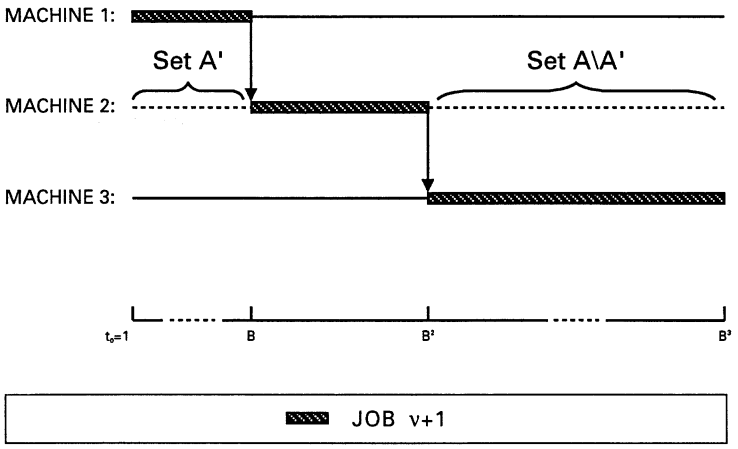

Fig. 3. A three-machine open-shop.

job exceeds time D). Without loss of generality, assume that these operations are scheduled on machines 1, 2 and 3 in this order, starting at time 1. Since there is no solution to **SP**, any subset A' of jobs processed on machine 2 prior to operation 2 of job $v+1$, will be completed strictly prior to time B. Since $\prod_{i\in A'} a_i < B \prod_{i\in A\setminus A'} a_i > B$. It follows that the completion time of the last job on machine 2 will be at $B^2 \prod_{i\in A\setminus A'} a_i > B^3 = D$. □

4. Job-shops

We assume now that each job is given its own ordering requirement, i.e. we consider job-dependent routes through the machines. This *job-shop* setting is clearly more complex than both flow-shop and open-shop settings. We prove NP-hardness even for case of two machines. We consider again "Subset Product" for the reduction.

The associated decision problem of makespan minimization on an M-machine job-shop with simple linear deterioration is the following:

JS: Given a set $J = \{1,2,\ldots,N\}$ of jobs, an M-machine job-shop ($M \in Z^+$), a deterioration rate $\alpha_{ij} \in R^+ \cup \{0\}$ of job i on machine j ($i = 1,\ldots,N, j = 1,\ldots,M$), a common release date t_0, and a positive integer D, is there a schedule q such that $C_{\max}(q) \leqslant D$?

Theorem 5. *Problem* **JS** *is* NP*-complete even for 2 machines.*

Proof. Clearly **JS** $\in$ NP. Let I_{SP} be an instance of **SP**. We construct the corresponding instance I_{JS} of **JS**. Let $N = v + 1$ and $M = 2$.

Again we denote $\alpha_{i1} = \alpha_i,\ \alpha_{i2} = \beta_i,\ i = l,\ldots,v$.

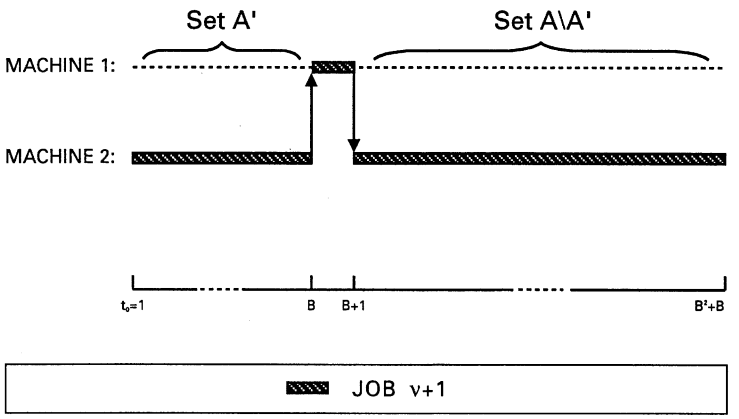

Fig. 4. A two-machine job-shop.

Each one of the first v jobs consists of a single operation which needs to be done on machine 1. Deterioration rates for these jobs are given by $\alpha_i = a_i - 1$, $i = 1, \ldots, v$. Job $v+1$ consists of three operations with the following required order: the first is to be done on machine 2, the second is to be done on machine 1 and the third is to be done on machine 2. Deterioration rates are:

$$\alpha_{v+1} = \frac{1}{B}, \quad \beta_{v+1} = B - 1.$$

$$t_0 = 1 \quad \text{and} \quad D = (B+1)B.$$

As in the previous cases, this construction can be accomplished in polynomial time.

Following Theorems 2 and 4 we prove the following two statements:

(i) If for instance I_{SP}, there exists a subset A' such that $\prod_{i\in A'} a_i = \prod_{i\in A\setminus A'} a_i = \sqrt{\prod_{i\in A} a_i}$, then for instance I_{JS} there exists a schedule q such that $C_{\max}(q) \leqslant D$.

For the given solution of I_{SP}, schedule the jobs as follows: Schedule the first operation of job $v+1$ on machine 2 at time 1. This operation will be completed at time B. Then start the second operation of job $v+1$ on machine 1 at time B. Upon completion of this operation (at time $B+1$), start the third operation of job $v+1$ on machine 2. This operation will be completed at time $D = (B+1)B$. We are left with the v jobs to be scheduled on machine 1. Schedule the jobs in set A' to start at time 1 (and finish at time B), and schedule the jobs in set $A \setminus A'$ to start at time $B+1$ and finish at $(B+1)B$. This schedule has the required makespan; see Fig. 4.

(ii) If there is no solution for **SP**, then there is no solution for **JS**. Similar to Theorem 2, any optimal schedule must have no idle time between the operations of job $v+1$. Thus, these operations must start as soon as possible (i.e. the first operation must start on machine 2 at time 1, the second operation must start on machine 1 at time B, and the third operation must start on machine 2 at time $B+1$). Since there is no solution to **SP**, any subset A' of jobs processed on machine 1 prior to the second operation

of job $v+1$, will be completed strictly prior to time B. $\prod_{i\in A\setminus A'} a_i > \sqrt{\prod_{i\in A} a_i} = B$, and therefore the completion time of the last job on machine 1 will be at $(B+1)\prod_{i\in A\setminus A'} a_i > (B+1)B = D$. □

5. Conclusion

We provided a complete analysis of shop scheduling problems with linearly deteriorating jobs. We showed that the two-machine flow-shop and the two-machine open-shop are polynomially solvable, but the three-machine flow-shop, the three-machine open-shop and the two-machine job-shop are already NP-hard. An open question remains whether pseudo-polynomial algorithms exist for the latter cases. Recall that the classical $F_3//C_{\max}$ and the classical $J_2//C_{\max}$ are strongly NP-hard (see [5,10], respectively), but the classical $O_3//C_{\max}$ is NP-hard in the ordinary sense [6]. Further research may thus focus on investigating the analogous cases in the setting of deteriorating jobs. Another direction for future research would be the identification of special cases of shops with deteriorating jobs, which have polynomial time solutions. Finally, an analysis of the complexity status of the same shops studied in this paper with different objectives (e.g. minimizing the sum of job completion times), can be a challenge for a new line of research.

Acknowledgements

I want to thank Dina Sher for preparing the graphs. This paper was supported in part by the Recanati Fund of The School of Business Administration, The Hebrew University, Jerusalem, Israel.

References

[1] S. Browne, U. Yechiali, Scheduling deteriorating jobs on a single processor, Oper. Res. 38 (1990) 495–498.
[2] Z-L. Chen, Parallel machine scheduling with time dependent processing times, Discrete Appl. Math. 70 (1996) 81–94.
[3] T.C.E. Cheng, Q. Ding, A state-of-the-art survey on scheduling with time-dependent processing times, Working Paper, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, 1998.
[4] M.R. Garey, D.S. Johnson, Computers and Intractability — A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, San Francisco, 1979.
[5] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flowshop and jobshop scheduling, Math. Oper. Res. 1 (1976) 117–129.
[6] T. Gonzalez, S. Sahni, Open shop scheduling to minimize finish time, J. Assoc. Comput. Mach. 23 (1976) 665–679.
[7] S.M. Johnson, Optimal two- and three-stage production schedules with setup times included, Naval Res. Logist. Quart. 1 (1954) 61–67.
[8] D.S. Johnson, The NP-completeness column: an ongoing guide, J. Algorithms 4 (1981) 393–405.
[9] W. Kubiak, S.L. van de Velde, Scheduling deteriorating jobs to minimize makespan, Naval Res. Logist. 45 (1998) 511–523.

[10] J.K. Lenstra, A.H.G. Rinnooy Kan, Computational complexity of discrete optimization problems, Ann. Discrete Math. 4 (1979) 121–140.
[11] G. Mosheiov, V-Shaped policies to schedule deteriorating jobs, Oper. Res. 39 (1991) 979–991.
[12] G. Mosheiov, Scheduling deteriorating jobs under simple linear deterioration, Comput. Oper. Res. 21 (1994) 653–659.
[13] G. Mosheiov, On flow-shop scheduling with deteriorating jobs, Working Paper, The Hebrew University, Jerusalem, 1996.
[14] G. Mosheiov, Multi-machine scheduling with linear deterioration, INFOR 36 (1998) 205–214.
[15] P.S. Sundararaghavan, A.S. Kunnathur, Single machine scheduling with start time dependent processing times: some solvable cases, European J. Oper. Res. 78 (1994) 394–403.