Journal of the Operational Research Society (2014) 65, 151–157

www.palgrave-journals.com/jors/

Polynomial time solutions for scheduling problems on a proportionate flowshop with two competing agents

B Mor[1] and G Mosheiov[2*]

[1]*The Department of Economics and Business Administration, Ariel University Center of Samaria, Israel; and* [2]*The Hebrew University, Jerusalem, Israel*

In scheduling problems with two competing agents, each one of the agents has his own set of jobs and his own objective function, but both share the same processor. The goal is to minimize the value of the objective function of one agent, subject to an upper bound on the value of the objective function of the second agent. In this paper we study two-agent scheduling problems on a proportionate flowshop. Three objective functions of the first agent are considered: minimum maximum cost of all the jobs, minimum total completion time, and minimum number of tardy jobs. For the second agent, an upper bound on the maximum allowable cost is assumed. We introduce efficient polynomial time solution algorithms for all cases.

Journal of the Operational Research Society (2014) **65,** 151–157. doi:10.1057/jors.2013.9
Published online 27 February 2013

Keywords: two-agent scheduling; proportionate flowshop; makespan; total completion time; number of tardy jobs

1. Introduction

Baker and Smith (2003) and Agnetis *et al* (2004) started a new line of research in scheduling theory, known as *scheduling with two competing agents*. The general setting of a problem in this class consists of two agents who need to process their jobs using the same processor. Each one of the two agents has his own set of jobs and his own objective function. The goal is to minimize the value of the objective function of one agent, subject to an upper bound on the value of the objective function of the second agent.

A growing number of papers have been published in recent years focusing on scheduling with two competing agents. These papers assume various combinations of scheduling measures and machine settings. The relevant reference list contains Baker and Smith (2003) (focused on a single machine and the criteria of makespan, maximum lateness and total weighted completion time), Agnetis *et al* (2004) (considered the same and additional measures, eg the number of tardy jobs and maximum regular functions, and studied multi-machine settings as well), Cheng *et al* (2006) (proved that the problem with minimum number of tardy jobs for each agent is strongly NP-hard, and introduced a polynomial time solution for the case of unit time jobs), Ng *et al* (2006) (studied minimum weighted completion time for one agent subject to an upper bound on the number of tardy jobs for the second agent), Cheng *et al* (2007) (studied total tardiness), Agnetis *et al* (2007) (focused on a multi-agent setting), Cheng *et al* (2008) (multi-agent problems with precedence constraints), Liu and Tang (2008) and Liu *et al* (2009) (two agent scheduling with deteriorating jobs), Agnetis *et al* (2009) (branch-and-bound algorithms for minimum total weighted completion time of the first agent, subject to an upper bound on several measures of the second agent), Lee *et al* (2009) (introduced multi-agent scheduling with total weighted completion time, and provided fully polynomial approximation schemes), Mor and Mosheiov (2010) (two-agent scheduling with various earliness measures), Leung *et al* (2010) (two-agent scheduling with preemption and release dates), Wan *et al* (2010) (two-agent problems with controllable processing times), Liu *et al* (2010) (two-agent single-machine scheduling with position-dependent processing times), Cheng *et al* (2011) (minimum weighted completion time of the first agent subject to no tardy jobs of the second agent, with learning effect based on sum-of-processing times), Li and Yuan (2012) (two-agent scheduling with batching, where the total processing time of a batch is equal to the maximum processing time of the jobs in the batch), Mor and Mosheiov (2011) (two-agent batch scheduling assuming identical jobs and minimum total flowtime), Li and Hsu (2012) (minimizing total weighted completion time of both agents, subject to an upper bound on the makespan of both agents, with a learning effect), Gawiejnowicz *et al* (2011) (minimum total tardiness of the first agent subject

*Correspondence: G Mosheiov, School of Business Administration, The Hebrew University of Jerusalem; Mount Scopus, Jerusalem 91905, Israel.
E-mail: msomer@mscc.huji.ac.il

to no tardy jobs of the second agent, with time-dependent processing times), Gerstl and Mosheiov (2012) (minimizing weighted earliness-tardiness of the first agent, subject to maximum weighted deviation of the second agent), Yin *et al* (2012a) (minimizing earliness measures of the first agent subject to an upper bound on the earliness of the second agent, with job linear deterioration), Yin *et al* (2012b) (minimum tardiness of the first agent subject to minimum lateness of the second agent, with release dates), and Yin *et al* (2012c) (two-agent single machine with release times and deadlines).

The vast majority of the above papers consider a single-machine setting. Among the papers dealing with multi-machine settings, only Agnetis *et al* (2004) studied flowshops. They focused on the simplest case, that is, on a two-machine flowshop with the makespan objective function for both agents. They proved that even this case is binary NP-hard. In this paper we further investigate scheduling problems with two competing agents on flowshops. We focus on the well-known special case of *proportionate flowshop*, in which the job processing times are *machine-independent*. We consider the following classical objective functions of the first agent: (i) minimizing the maximum cost among all the jobs (given job-dependent general cost functions), (ii) minimizing the total completion times of all the jobs, and (iii) minimizing the number of tardy jobs. In all cases, a maximum allowable cost on each of the jobs of the second agent is assumed. We introduce polynomial time solutions for all three problems.

The paper is organized as follows: In Section 2 we provide the problem formulation. In Sections 3–5, we solve the minmax cost problem, the total completion time problem and the number of tardy jobs problem, respectively.

2. Formulation

Two agents, denoted X and Y, need to process n^X and n^Y jobs, respectively, on an m-machine flowshop. Let $n \equiv n^X + n^Y$. The processing time of job j of agent A on machine i is denoted p_{ij}^A, $i=1, \ldots, m$, $j=1, \ldots, n^A$, $A=X, Y$. We assume a *proportionate* flowshop, that is, $p_{ij}^A = p_j^A$, $i=1, \ldots, m$, $j=1, \ldots, n^A$, $A=X, Y$. All the jobs are available at time zero, and preemption is not allowed.

For a given job schedule, let C_j^A denote the completion time of the last operation (ie, on the last machine) of job j of agent A, $j=1, \ldots, n^A$, $A=X, Y$. Each job (of each agent) has a specific cost function that depends on its completion time. We denote these functions: f_j^A, $j=1, \ldots, n^A$, $A=X, Y$, and assume that they are general non-decreasing in the job completion time. Let $f_{\max}^A = \max\{f_1(C_1), f_2(C_2), \ldots, f_{n^A}(C_{n^A})\}$, $A=X, Y$ The first problem solved here is minimum $f_{\max}^X$ subject to an upper bound Q on the maximum cost of agent Y, $f_{\max}^Y$. Formally, the problem is $F_m/p_{ij}=p_j/f_{\max}^X: f_{\max}^Y \leqslant Q$.

The second problem studied here is minimum total flowtime of agent X, subject to an upper bound on the maximum cost of agent Y. The total flowtime of agent X is given by $\sum_{j=1}^{n^X} C_j^X$. The problem is $F_m/p_{ij}=p_j/\sum_{j=1}^{n^X} C_j^X : f_{\max}^Y \leqslant Q$.

For the third problem, we define due-dates for the X-jobs: d_j^X, $j=1, \ldots, n^X$. Let $U_j^X=0$ if $C_j^X \leqslant d_j^X$, and $U_j^X=1$ otherwise. Our objective is to minimize the number of tardy jobs of agent X, subject to an upper bound on the maximum cost of agent Y: $F_m/p_{ij}=p_j/\sum_{j=1}^{n^X} U_j^X : f_{\max}^Y \leqslant Q$.

3. $F_m/p_{ij}=p_j/f_{\max}^X: f_{\max}^Y \leqslant Q$.

We begin by solving the single-agent version of this problem, that is, $F_m/p_{ij}=p_j/f_{\max}$. We show that this problem is solved by Lawler's algorithm (1973), which was designed for a single-machine setting. Let n denote the number of jobs (of a single agent), p_j is the processing time of job j, and $p_{\max}=\max\{p_j, j=1, \ldots, n\}$. Let f_j denote the cost function of job j, which is assumed to be general non-decreasing in its completion time. Also, let T denote the completion time of the last job on the last machine (ie the makespan value). It is well-known that T is sequence-independent, and is given by: $T=(m-1)p_{\max}+\sum_{j=1}^{n} p_j$ (see eg, Pinedo, 2012). We define k to be the index of the job with the minimal cost function, if scheduled to be completed at time T, that is, $k=\arg\{\min\{f_i(T)), j=1, \ldots, n\}\}$.

Claim 3.1 *An optimal schedule exists in which job k is scheduled last.*

Proof As in Lawler (1973), consider an optimal schedule S in which job k is not scheduled last. Thus, S consists of a subset of jobs q_1 followed by job k, followed by a subset q_2, followed by job l, the last job in S; see Figure 1(a). We create a sequence S' by moving job k to be last (immediately after l; see Figure 1(b)). We claim that S' is at least as good as S. First, the completion times of the jobs of q_1 on the last machine of the flowshop in S' are clearly identical to their completion times in S, implying that their costs are identical in both sequences. Next, it is easily observed that in a proportionate flowshop, jobs completed earlier on the first machine will be completed earlier on the last machine as well. Thus, the completion time of job l and the completion times of the jobs of q_2 in S' (on the last machine of the flowshop) are not larger than their completion times in S. This implies that their costs are not larger (due to the monotonicity of the cost functions). Finally the cost of the last job in S' (job k) is not larger than the cost of the last job in S (job l), due to the definition of k. It follows that sequence S' is optimal, which completes the proof. □

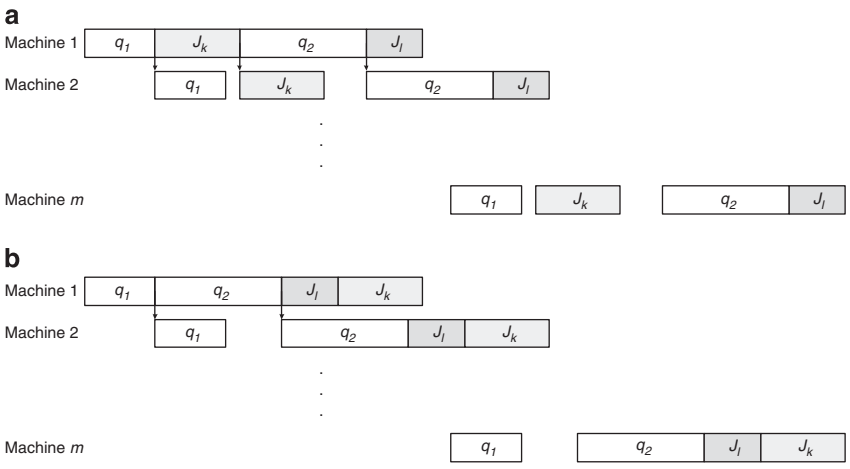

Figure 1 (a) Schedule S in Claim 3.1; (b) Schedule S' in Claim 3.1.

Claim 3.1 verifies that Lawler's Algorithm for a single machine produces an optimal schedule for a proportionate flowshop as well. The modified algorithm is the following:

Lawler Algorithm for proportionate flowshop:

Input: an m-machine proportionate flowshop,
a set q of n jobs with processing times $\{p_1, p, \ldots, p_n\}$ and cost functions $\{f_1, f, \ldots, f_n\}$.
Step 1: Calculate $T=(m-1)p_{\max}+\sum_{j=1}^{n}p_j$;
Step 2: Calculate $f_j(T)$, $j\in q$;
Find $k=\arg\{\min\{f_j(T), j\in q\}\}$;
Step 3: Schedule job k in the last position.
$q=q\backslash\{k\}$;
If $q=\varnothing$
Stop;//The resulting schedule is optimal.
$p_{\max}=\max\{p_j, j\in q\}$;
$T=(m-1)p_{\max}+\sum_{j\in q}p_j$;
Go to Step 2;

Running time: Assuming that calculating the f_j values is done in constant time, Step 2 requires $O(n)$, and is performed n times. Hence the total running time is $O(n^2)$.

Claim 3.2 *The two-agent version can be reduced to a single-agent problem.*

Proof Following Agnetis *et al* (2004), we calculate the deadlines of the jobs of agent Y, that is, $D_j^Y=\max\{t; f_j^Y(t)\leqslant Q\}$, $j=1, \ldots, n^Y$. (Note that the completion time of a job is defined as the completion time of its last operation. Hence, D_j^Y is the latest permitted completion time of job j on machine m. Its latest permitted completion time on the first machine of the proportionate flowshop is clearly $D_j^Y-mp_j^Y$.) We also define q^X and q^Y to be the sets of jobs of agents X and Y, respectively. Given these, the reduction to a single-agent problem is based on the following definition of the cost function:

$$f(x)=\begin{cases} f_j^X(t), & j\in q^X \\ \infty, & j\in q^Y \text{ and } t>D_j^Y \\ 0, & j\in q^Y \text{ and } t\leqslant D_j^Y \end{cases} \quad (1)$$

This resulting single-agent setting is clearly solved to optimality by the above algorithm. □

Based on the above two claims, we have the following algorithm:

Algorithm for $F_m/p_{ij}=p_j/f_{\max}^X$: $f_{\max}^Y\leqslant Q$

Input: An m-machine proportionate flowshop, two agents X and Y;
Two sets of jobs q^A, $A=X, Y$;
The processing times and cost functions: p_j^A, f_j^A, $j=1, \ldots, n^A$, $A=X,Y$.
Step 1: Let $q=q^X\cup q^Y$; $p_{\max}=\max\{p_j, j\in q\}$; $T=(m-1)p_{\max}+\sum_{j\in q}p_j$;
Define the functions f_j, $j\in q$, according to (1).
Step 2: Calculate D_j^Y, $j=1, \ldots, n^Y$;
Step 3: Sort D_j^Y in a non-decreasing order;
Step 4: Use *Lawler Algorithm for proportionate flowshop* (see above) to obtain an optimal solution. (If $q^X=\varnothing$ and $f_j^Y(t)>Q$, no feasible solution exists.)

Running time: We assume that calculating the f_j functions and their inverse functions is done in constant time. Then, calculating the D_j^Y values (Step 2) requires $O(n)$. Sorting the Y-jobs by their deadlines (Step 3) requires $O(n \log n)$.

As mentioned above, Lawler Algorithm (Step 4) requires $O(n^2)$. Thus, the total running time is $O(n^2)$.

[*Comment*: We have programmed and tested the above algorithm. The C++ program was processed on Intel® Core™ i7-2600k@3.40GHz. We solved a three-machine proportionate flowshop with 5000 jobs for each agent. For simplicity we assumed for the jobs of agent X: $f_j^X = C_j$, $j=1, \ldots, n^X$. For the jobs of agent Y, we considered step functions: $f_j^Y(C)=0$ if $C \leqslant Q_j$, (for some job-dependent constant Q_j), and $f_j^Y(C)=\infty$ if $C > Q$, $j=1, \ldots, n^Y$. The job processing times were generated uniformly in the interval [1,100], and the job-dependent Q_j values of agent Y were generated uniformly in the interval $[3P_{\max}, C_{\max}]$, where $3P_{\max}$ is the completion time of the largest job on the third machine if this job is scheduled first, and $C_{\max}$ is the maximum completion time of all the jobs of both agents. The total running time required for solving this 10 000-job problem was 842 milliseconds, ie, less than a second.]

4. $F_m/p_{ij}=p_j/\sum C_j^X: f_{\max}^Y \leqslant Q$.

In this section we solve the problem of minimizing total completion time of agent X, subject to an upper bound on the maximum cost of agent Y. Recall that q denotes the set of all jobs (of both agents), $p_{\max}$ is the maximal processing time in the set q, and T denotes the (sequence independent) makespan obtained by all the jobs in q. Following Agnetis *et al* (2004) (for the single-machine case), we claim the following (for a proportionate flowshop):

Claim 4.1 *If a Y-job, say k, exists such that $f_K^Y(T) \leqslant Q$, then an optimal schedule exists in which job k is scheduled last (to be completed at T).*

Proof Consider an optimal schedule S in which the Y-job k is not scheduled last. S consists of a subset of jobs q_1 followed by job k, followed by a subset q_2, followed by an X-job which is scheduled last. We create a sequence S' by moving job k to be last (and shifting the jobs of q_2 and the last scheduled X-job to start earlier). Note that the completion time of job k on the last machine in S' is identical to the completion time of the last X-job in S (by the fact that the makespan in a proportionate flowshop is sequence independent). Note also that the completion times of (all the operations of) the jobs in the subset q_2 and that of the last X-job were reduced on all the machines. We claim that: (i) the resulting schedule is clearly feasible (by the definition of k, and the fact that the completion time of all other Y-jobs was not increased), (ii) the cost of the resulting schedule is smaller (since the completion time of some of the X-jobs decreases, while the completion time of the remaining jobs was not affected). Therefore, S' has a strictly smaller cost, implying that S is not optimal. □

Based on the above claim, an algorithm can be introduced in which the Y-jobs will be assigned as late as possible, and will be scheduled according to their deadlines. (Note that if several Y- jobs can be scheduled at time T, their order is in fact immaterial.)

Claim 4.2 *If for all the Y-jobs, $f_j^Y(T) > Q$, then an optimal schedule exists in which the largest X-job is scheduled last (to be completed at T).*

Proof By a standard pair-wise interchange argument. □

Based on this claim, an optimal schedule exists in which the X-jobs are scheduled according to SPT. A formal algorithm is provided below:

Algorithm for $F_m/p_{ij}=p_j/\sum C_j^X: f_{\max}^Y \leqslant Q$

Input: An m-machine proportionate flowshop, two agents X and Y.
Two sets of jobs q^A, $A=X,Y$.
The processing times: p_j^A, $j=1, \ldots, n^A$, $A=X, Y$.
The cost functions: f_j^Y, $j=1, \ldots, n^Y$.
Step 1: Calculate D_j^Y, $j=1, \ldots, n^Y$;
Step 2: Sort D_j^Y in a non-decreasing order;
Step 3: Sort p_j^X in a non-decreasing order;
$j^X=n^X$; $j^Y=n^Y$; $\sum C_j=0$;
Step 4: While ($q^X \neq \varnothing$ and $q^Y \neq \varnothing$)
$p_{\max}^X=\max\{p_j^X, j\in q^X\}$; $p_{\max}^Y=\max\{p_j^Y, j\in q^Y\}$;
$p_{\max}=\max\{p_{\max}^X, p_{\max}^Y\}$;
$T=(m-1)p_{\max}+\sum_{j\in q^X} p_j^X+\sum_{j\in q^Y} p_j^Y$;
If $q^Y \neq \varnothing$
If $f_{j^Y}^Y(T) \leqslant Q$, then
schedule job j^Y to be completed at time T; $q^Y=q^Y\setminus\{j^Y\}$; $j^Y=j^Y-1$;
Else,
If $q^X \neq \varnothing$, then
schedule job j^X to be completed at time T; $\sum C_j=\sum C_j+T$; $j^X=j^X-1$;
Else—no feasible solution exists;
Else
If $q^X \neq \varnothing$, then
schedule job j^X to be completed at time T; $\sum C_j=\sum C_j+T$; $j^X=j^X-1$;
Endif
Endif
Endwhile

Running time: Step 1 requires $O(n)$ time (assuming that calculating the D_j^Y values and the inverse functions is done in $O(1)$). Sorting the Y-jobs by their due-dates (Step 2) requires $O(n \log n)$. Sorting the X-jobs according to their

processing times (Step 3) requires $O(n \log n)$. Each iteration in Step 4 requires $O(1)$, implying that Step 4 is performed in $O(n)$ We conclude that the running time of the entire algorithm is $O(n \log n)$.

5. $F_m/p_{ij}=p_j/\sum U_j^X: f_{\max}^Y \leqslant Q$

In this section we assume that the X-jobs have due-dates, denoted by d_j^Y, and introduce a polynomial time solution for the problem of minimizing the number of tardy jobs of agent X, subject to an upper bound on the maximal cost of agent Y. We show that the solution introduced by Agnetis *et al* (2004) for the single-machine case can be easily extended to a proportionate flowshop setting.

Recall that the deadlines for the jobs of agent Y are denoted by $D_j^Y=\max\{t; f_j^Y(t)\leqslant Q\}$, $j=1, \ldots, n^Y$. As mentioned, a deadline is clearly defined as the latest possible completion time of the last operation of the job (ie on the last machine of the flowshop). For convenience, we denote the Latest permitted Completion time of a Y-job j on the *first* machine by $LC_j^{(Y,m1)}$. Similarly, we denote the Latest permitted Starting time of a Y-job j on the *first* machine by $LS_j^{(Y,m1)}$. A trivial upper bound on $LC_j^{(Y,m1)}$ is $D_j^Y-(m-1)p_j^Y$. Another upper bound on $LC_j^{(Y,m1)}$ is the latest permitted starting of job $j+1$ on the first machine ($LS_{j+1}^{(Y,m1)}$). It follows that $LC_j^{(Y,m1)}\leqslant \min\{D_j^Y-(m-1)p_j^Y, LS_{j+1}^{(Y,m1)}\}$, and consequently $LS_j^{(Y,m1)}\leqslant \min\{D_j^Y-mp_j^Y, LS_{j+1}^{(Y,m1)}-p_j^Y\}$. Based on these definitions, one can easily schedule the Y-jobs as late as possible: schedule the last Y-job to start processing at time $D_n^Y-mp_n^Y$ on the first machine, and continue backwards, such that the starting time of job j on the first machine is given by $LS_j^{(Y,m1)}=\min\{D_j^Y-mp_j^Y, LS_{j+1}^{(Y,m1)}-p_j^Y\}$.

Following Agnetis *et al* (2004), we schedule the Y-jobs at their latest times, and consider the resulting intervals as *unavailable intervals* for the X-jobs. We then create an instance for the problem of the single agent X only ($1//\sum U_j^X$) as follows: (i) we remove the unavailable intervals from all the machines, and (ii) we re-define the due-dates of the X-jobs, by subtracting the cumulative unavailable intervals prior to each due-date. Formally, let T_j^X denote the total processing time of the Y-jobs processed prior to d_j^X, $j=1, \ldots, n_X$. Then the updated due-dates of the X-jobs are: $d_j^{X'}=d_j^X-T_j^X$, $j=1, \ldots, n_X$. Note that the due-dates are updated similarly on all the machines. In particular, the due-date of the first operation (on the first machine) of job j becomes $d_j^{(X',m1)}=d_j^{(X,m1)}-T_j^X=d_j^X-(m-1)\ p_j-T_j^X$, $j=1, \ldots, n_X$. The single-agent problem $1/d_j^{X'}/\sum U_j^X$ is known to be solved by Moore's Algorithm (Moore, 1968), even for a proportionate flowshop (Pinedo, 2012). After the problem $1/d_j^{X'}/\sum U_j^X$ is solved to optimality, the Y-jobs are re-inserted on all the machines at their latest starting times (ie the unavailable intervals are reconsidered). The resulting schedule may clearly include some preempted X-jobs. We note that the resulting number of tardy X-jobs is clearly a lower bound on the optimal number of tardy jobs for the non-preemptive problem. However, similar to the single-machine case (see Agnetis *et al*, 2004), this schedule can be easily converted into a schedule with no preemption and the same number of tardy X-jobs. Denote by $S_j^{X'}$ the starting time of job j of agent X on the last machine. It follows that this job starts at time $S_j^{X'}-p_j^X$ on the previous machine, and accordingly at time $S_j^{(X',m1)}=S_j^{X'}-(m-1)p_j^X$ on the first machine. If an X-job j is preempted by one or more Y-jobs, then these Y-jobs (ie all operations of each of these Y-jobs) can be moved to start earlier. The first of these Y-jobs will start at time $S_j^{(X',m1)}$ on machine 1, and at time $S_j^{X'}$ on the last machine. All the Y-jobs which preempted job j of agent X are processed continuously. As a result, the X-operations start later and processed continuously. Note that in the resulting schedule: (i) the Y-jobs are completed earlier implying that the schedule remains feasible, (ii) the X-job is not preempted, and (iii) the completion time of the X-job has not changed. Repeating this procedure for all the preempted X-jobs leads to a feasible schedule, that is, no Y-jobs are late and no X-jobs are preempted. The resulting number of tardy jobs remains equal to the above lower bound, implying that this solution is optimal. A more formal description of these steps is given in the following:

Algorithm for $F_m/p_{ij}=p_j/\sum U_j^X: f_{\max}^Y\leqslant Q$

Input: An m-machine proportionate flowshop, two agents X and Y.
Two sets of jobs q^A, $A=X,Y$.
The processing times: $p_j^A, j=1, \ldots, n^A$, $A=X,Y$.
The cost functions: $f_j^Y, j=1, \ldots, n^Y$.
The due-dates of the X-jobs: $d_j^X, j=1, \ldots, n^X$.

Step 1: Calculate $D_j^Y, j=1, \ldots, n^Y$;

Step 2: //Schedule the Y-jobs.
Sort D_j^Y in a non-decreasing order, and renumber the Y-jobs accordingly;
Schedule the last Y-job to start processing on machine 1 at time $D_n^Y-mp_n^Y$;
For $j=n^Y-1, \ldots, 1$
$LS_j^{(Y,m1)}=\min\{D_j^Y-mp_j^Y, LS_{j+1}^{(Y,m1)}-p_j^Y\}$;
//Schedule job j to start processing on machine 1 at time $LS_j^{(Y,m1)}$.
Endfor

Step 3: //Create the single-agent X problem.
Remove the processing times of the Y-jobs (unavailable intervals) from the problem;
Compute $T_j^X, j=1, \ldots, n_X$;//the total processing time of the Y-jobs processed prior to d_j^X.
Update the due-dates of the X-jobs: $d_j^{X'}=d_j^X-T_j^X, j=1, \ldots, n_X$;

Step 4: Solve the single-machine problem $1/d_j^{X'}/\sum U_j^X$;

Step 5: Reinsert the Y-jobs at their latest possible times on all the machines;
//This may lead to preemption of some X-jobs.

Step 6: For each preempted X-job, shift the relevant Y-jobs to start at time $S_j^{(X,\mathrm{m1})}$) on machine 1, and processed continuously;
//The starting times of the X-jobs are delayed on all the machines, they are processed non-preemptively and their completion times are unchanged.

Running time: If the computation of the f_j values requires constant time, Step 1 requires $O(n)$. and is performed n times. Hence the total running time is $O(n^2)$. Sorting the D_j^Y requires $O(n \log n)$, and scheduling the Y-jobs takes $O(n)$ time. Hence, Step 2 requires $O(n \log n)$. Step 3 is performed in $O(n)$, and Step 4 needs $O(n \log n)$. Steps 5 and 6 are performed in $O(n)$ as well. Thus, the total running time is $O(n \log n)$.

6. Conclusion

We studied scheduling problems with two competing agents on a proportionate flowshop. Three objective functions of the first agent were considered: minimum maximum cost, minimum total completion time, and minimum number of tardy jobs. In all cases, a maximum allowable cost on the jobs of the second agent is assumed. The two-agent problem on a flowshop was shown to be NP-hard even for the setting of two machines and for the simplest objective functions of makespan minimization for both agents (Agnetis *et al*, 2004). However, problems in the setting of proportionate flowshop studied in this paper are shown to have polynomial time solutions. Following the single-machine solutions introduced by Agnetis *et al* (2004), we propose $O(n^2)$, $O(n \log n)$ and $O(n \log n)$ algorithms for the above three problems, respectively. Future research may focus on extensions of the two-agent proportionate flowshop model to other objective functions of both agents.

Acknowledgements—This paper was supported in part by The Charles Rosen Chair of Management and the Recanati Fund of The School of Business Administration, The Hebrew University, Jerusalem, Israel.

References

Agnetis A, Mirchandani PB, Pacciarelli D and Pacifici A (2004). Scheduling problems with two competing agents. *Operations Research* **52**(2): 229–242.

Agnetis A, Mirchandani PB, Pacciarelli D and Pacifici A (2007). Multi-agent single machine scheduling. *Annals of Operations Research* **150**(1): 3–15.

Agnetis A, Pascale G and Pacciarelli D (2009). A Lagrangian approach to single-machine scheduling problems with two competing agents. *Journal of Scheduling* **12**(4): 401–415.

Baker KR and Smith JC (2003). A multiple-criterion model for machine scheduling. *Journal of Scheduling* **6**(1): 7–16.

Cheng TCE, Ng CT and Yuan JJ (2006). Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs. *Theoretical Computer Science* **362**(1–3): 273–281.

Cheng TCE, Ng CT and Yuan JJ (2007). *Two-agent scheduling on a single machine with fixed jobs and preemption*. Working paper, Department of Logistics, Hong Kong Polytechnic University.

Cheng TCE, Ng CT and Yuan JJ (2008). Multi-agent scheduling on a single machine with max-form criteria. *European Journal of Operational Research* **188**(2): 603–609.

Cheng TCE, Cheng S-R, Wu W-H, Hsu P-H and Wu C-C (2011). A two-agent single-machine scheduling problem with truncated sum-of-processing-times-based learning considerations. *Computers & Industrial Engineering* **60**(4): 534–541.

Gawiejnowicz S, Lee W-C, Lin C-L and Wu C-C (2011). Single-machine scheduling of proportionally deteriorating jobs by two agents. *Journal of the Operational Research Society* **62**(11): 1983–1991.

Gerstl E and Mosheiov G (2012). Scheduling problems with two competing agents to minimize weighted earliness-tardiness. *Computers & Operations Research* **40**(1): 109–116.

Lawler EL (1973). Optimal sequencing of a single machine subject to precedence constraints. *Management Science* **19**(5): 544–546.

Lee K, Choi B-C, Leung JY-T and Pinedo ML (2009). Approximation algorithms for multi-agent scheduling to minimize total weighted completion time. *Information Processing Letters* **109**(16): 913–917.

Leung JY-T, Pinedo M and Wan G (2010). Competitive two-agent scheduling and its applications. *Operations Research* **58**(2): 458–469.

Li D-C and Hsu P-H (2012). Solving a two-agent single-machine scheduling problem considering learning effect. *Computers & Operations Research* **39**(7): 1644–1651.

Li S and Yuan J (2012). Unbounded parallel-batching scheduling with two competitive agents. *Journal of Scheduling* **15**(5): 629–640.

Liu P and Tang L (2008). Two-agent scheduling with linear deteriorating jobs on single machine. In *Proceedings of the 14th Annual International Conference on Computing and Combinatorics*, pp 642-650, Dalian, China.

Liu P, Tang L and Zhou X (2009). Two-agent group scheduling with deteriorating jobs on a single machine. *The International Journal of Advanced Manufacturing Technology* **47**(5–8): 657–664.

Liu P, Zhou X and Tang L (2010). Two-agent single-machine scheduling with position-dependent processing times. *The International Journal of Advanced Manufacturing Technology* **48**(1–4): 325–331.

Moore JM (1968). An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science* **15**(1): 102–109.

Mor B and Mosheiov G (2010). Scheduling problems with two competing agents to minimize minmax and minsum earliness measures. *European Journal of Operational Research* **206**(3): 540–546.

Mor M and Mosheiov G (2011). Single machine batch scheduling with two competing agents to minimize total flowtime. *European Journal of Operational Research* **215**(3): 524–531.

Ng CT, Cheng CTE and Yuan JJ (2006). A note on the complexity of the two-agent scheduling on a single machine. *Journal of Combinatorial Optimization* **12**(4): 387–394.

Pinedo M (2012). *Scheduling: Theory, Algorithms, and Systems*. Springer: New York.

Wan G, Vakati SR, Leung JY-T and Pinedo M (2010). Scheduling two agents with controllable processing times. *European Journal of Operational Research* **205**(3): 528–539.

Yin Y, Cheng S-R and Wu C-C (2012a). Scheduling problems with two agents and a linear non-increasing deterioration to minimize earliness penalties. *Information Sciences* **189**: 282–292.

Yin Y, Wu W-H, Cheng S-R and Wu C-C (2012b). An investigation on a two-agent single-machine scheduling problem with unequal release dates. *Computers & Operations Research* **39**(12): 3062–3073.

Yin Y, Cheng S-R, Cheng TCE, Wu W-H and Wu C-C (2012c). Two-agent single-machine scheduling with release times and deadlines. *International Journal of Shipping and Transport Logistics* **5**(1): 75–94.

Received May 2012;
accepted January 2013 after two revisions