Ann Oper Res (2017) 253:227–245
DOI 10.1007/s10479-016-2310-5

ORIGINAL PAPER

Scheduling with two competing agents to minimize total weighted earliness

Enrique Gerstl[1,2] · **Baruch Mor**[3] · **Gur Mosheiov**[1]

Published online: 16 September 2016

Abstract We study a single machine scheduling problem with two competing agents and earliness measures. Given a common deadline for all the jobs of both agents, the objective function is minimizing the total weighted earliness of the first agent, subject to an upper bound on the maximum earliness of the jobs of the second agent. This problem was recently proved to be NP-hard, leaving the question of the complexity class open. We introduce a pseudo-polynomial dynamic programming algorithm, implying that the problem is NP-hard in the ordinary sense. An extensive numerical study indicates that the dynamic programming is very effective for solving medium size instances. We also propose an efficient heuristic, which is shown numerically to produce very close-to-optimal schedules. The dynamic programming algorithm is extended to any (given) number of agents, proving NP-hardness in the ordinary sense of the general multi-agent setting. Finally, we study the inverse problem of minimizing the maximum earliness of one agent subject to an upper bound on the maximum total weighted earliness of the second agent. We introduce a pseudo-polynomial dynamic programming algorithm, a simple greedy-type heuristic and a lower bound. Our numerical tests verify that the heuristic produces very small optimality gaps.

Keywords Two-agent scheduling · Single machine · Earliness · Dynamic programming

1 Introduction

The area of *multi-agent scheduling* deals with a setting of several agents (producers) using the same processor (machine). Each of the agents has a set of jobs, and each aims to minimize a scheduling measure depending on the completion times of his jobs. A growing number

✉ Enrique Gerstl
enrique.gerstl@mail.huji.ac.il

1 School of Business Administration, The Hebrew University, 91905 Jerusalem, Israel

2 School of Industrial Engineering, Jerusalem College of Technology, Jerusalem, Israel

3 Department of Economics and Business Administration, Ariel University, 40700 Ariel, Israel

of papers (starting with Baker and Smith 2003 and Agnetis et al. 2004), studying various models of multi-agent problems, have been published in the last decade. A list of representing papers contains: Agnetis et al. (2007), Wan et al. (2010), Mor and Mosheiov (2011), Li and Hsu (2012), Cheng et al. (2013), Donatas and T'kindt (2014), Gawiejnowicz and Suwalski (2014), and Oron et al. (2015). We refer the reader to a recently published book (Agnetis et al. 2014), which summarizes the current knowledge in this area.

In this paper we study a two-agent scheduling problem with *earliness* measures. It appears that very few researchers have studied multi-agent scheduling settings in a Just-In-Time environment (where the objective function consists of both earliness and tardiness). The short list of relevant references contains: Zuobao and Weng (2005), and Gerstl and Mosheiov (2013, 2014). According to Agnetis et al. (2014), only two published papers focused solely on earliness measures: Mor and Mosheiov (2010) studied two-agent problems, where one agent minimizes the maximum or the total (weighted) earliness, given an upper bound on the maximum earliness of the second agent; and Yin et al. (2012) studied similar problems with linear deterioration of the job processing times. More recently, Cheng (2014a) studied another version of the above problem with time-dependent job deterioration, and Cheng (2014b) focused on a setting where the second agent's earliness and tardiness are bounded. The following problem was studied in the former two papers: given a common deadline for all the jobs of both agents, the objective function is minimizing the total weighted earliness of the first agent subject to a common upper bound on the maximum earliness of the jobs of the second agent. Mor and Mosheiov (2010) proved that this problem is NP-hard, and the question whether the problem is NP-hard in the ordinary sense remained open.

We focus here on this two-agent problem. First, we prove that the problem is NP-hard in the ordinary sense by an introduction of a pseudo-polynomial dynamic programming (DP) algorithm. We perform an extensive numerical study in order to measure the running time of the DP as a function of the input size. Problems of 200 jobs (of each of the two agents) with job processing times bounded by 100 were solved in less than 37 s, utilizing no more than 3 GB. We also introduce an efficient heuristic, which produced very small optimality gaps: the average optimality gap, for example, for 200-job problems was 1.002. Then, we extend the setting to that of *multi-agents*. Mor and Mosheiov (2010) proved that the case of an *arbitrary* number of agents is NP-hard in the strong sense. We extend the above mentioned (pseudo-polynomial) DP to a setting of a *given* number of agents, implying that this more general setting remains NP-hard in the ordinary sense. In the last section, we study the inverse problem of minimizing the maximum earliness of one agent, subject to an upper bound on the maximum weighted earliness of the other agent. We introduce a pseudo-polynomial dynamic programming for this (NP-hard) problem as well. Our numerical tests indicate that the DP was not practical for large size problems (the running time required for solving a 50-job problem was about 30 s). We thus introduce a simple greedy-type heuristic and a lower bound. The heuristic was tested numerically against the tight lower bound, and was shown to produce very close-to-optimal schedules (e.g., the average optimality gap for 200-job problems was 1.002). Given the above results, we claim that the construction of the Pareto optimal set is obtained in pseudo-polynomial time as well.

It should be noted that there is some symmetry between the problem studied here (minimum total weighted earliness of one agent subject to an upper bound on the maximum earliness of the second agent), and the problem of minimizing total weighted job completion times of one agent subject to an upper bound on the maximum completion time of the second agent. The latter was proved by Agnetis et al. (2004) to be NP-hard. Kellerer and Strusevich (2010) introduced a pseudo-polynomial dynamic programming for the problem, and

designed a fully polynomial-time approximation scheme. Despite the symmetry, the solution procedures of the two problems appear to be quite different.

In Sect. 2 we introduce the formulation of both problems. Section 3 contains the DP algorithm, and a report on its numerical performance. In Sect. 4 we introduce the heuristic and the results of our numerical study. Section 5 addresses the general setting of multi-agents. Section 6 focuses on the inverse problem, and contains an introduction of the relevant DP, the greedy heuristic and the lower bound.

2 Formulation

Two agents, denoted A and B, share a common processor. Agent A has a set J^A of n^A jobs. Similarly, Agent B has a set J^B of n^B jobs. The processing time of job j in the sets J^A and J^B is denoted by p_j^A, $j = 1, \ldots, n^A$, and p_j^B, $j = 1, \ldots, n^B$, respectively. Let $P^A = \sum_{j=1}^{n^A} p_j^A$ $\left(P^B = \sum_{j=1}^{n^B} p_j^B\right)$ denote the total processing time of the jobs of Agent A (B). Also, let $p_{max}^A = max\left\{p_j^A, j = 1, \ldots, n^A\right\}$ and $p_{max}^B = max\left\{p_j^B, j = 1, \ldots, n^B\right\}$ denote the maximal processing time among all the A-jobs and the B-jobs, respectively. Following Mor and Mosheiov (2010), we assume a common deadline for all the jobs of both agents, denoted by D. (In order to guarantee feasibility we require $D \geq P^A + P^B$. Note that $D > P^A + P^B$ leads to an optimal schedule with a single idle time interval prior to the first job, and no idle time between consecutive jobs. We thus, assume for convenience that $D = P^A + P^B$, implying that all the jobs are processed continuously in the interval $[0, D]$.)

For a given job sequence, C_j^A $\left(C_j^B\right)$ denotes the completion time of job j of Agent A (B). The earliness of job $j \in J^A$ is given by $E_j^A = \max\left\{0, D - C_j^A\right\}$. Similarly, the earliness of job $j \in J^B$ is given by $E_j^B = \max\left\{0, D - C_j^B\right\}$. Let w_j^A denote the weight of job $j \in J^A$. The total weighted earliness of Agent A is given by $\sum_{j=1}^{n^A} w_j^A E_j^A$. Let $E_{max}^B = \max\left\{E_j^B, j = 1, \ldots, n^B\right\}$ denote the maximum earliness incurred by any of the B-jobs.

The first problem studied in this paper is, as mentioned above, of minimizing the total weighted earliness of the A-jobs, subject to an upper bound (U) on the maximum earliness value of the B-jobs. (We assume that $U < D - p_{max}^B$. Otherwise, a trivial solution is obtained by scheduling all the B-jobs first, followed by the A-jobs. On the other hand, in order to guarantee feasibility $U \geq P^B - p_{max}^B$.) Formally, we solve the following problem:

$$1/d_j = D(deadline)/\sum_{j=1}^{n^A} w_j^A E_j^A : E_{max}^B.$$

The second problem studied here is the inverse problem, i.e., that of minimizing the maximum earliness of Agent B subject to an upper bound (U) on the total weighted earliness of Agent A:

$$1/d_j = D\,(deadline)/E_{max}^B : \sum_{j=1}^{n^A} w_j^A E_j^A.$$

3 A dynamic programming algorithm for $1/d_j = D(deadline)/\sum_{j=1}^{n^A} w_j^A E_j^A : E_{max}^B$

We start by introducing several properties of an optimal schedule. The proofs are provided in "Appendix 1".

Property 1 An optimal schedule exists in which all the jobs of Agent B are scheduled (continuously) in a single block.

Property 2 An optimal schedule exists in which the largest B-job is scheduled first in the block of the B-jobs.

Property 3 An optimal schedule exists in which the A-jobs are scheduled in (at most) two blocks, and the jobs in each block are ordered in a non-increasing order of p_j^A/w_j^A.

Based on these properties, we propose in the following a pseudo-polynomial dynamic programming (DP) solution algorithm. We assume first that the starting time of the block of the B-jobs (denoted by t) is given. The completion time of the B-block (by Property 1) is $t + P^B$. We now specify all the relevant t values. Recall that U is the maximum allowable earliness value on the B-jobs. Thus, let the first scheduled B-job be completed at time $D-U$. Given Property 2, we assume that the largest B-job (whose processing was denoted p_{max}^B) is scheduled first. This case clearly reflects the minimal possible starting time of the B-block: $t_{min} = D - U - p_{max}^B$ (since any schedule containing a B-block starting prior to t_{min} violates the upper bound constraint.) It follows that $t \geq t_{min}$. The maximum possible t value is specified in the following:

Property 4 An upper bound on the starting time t of the B-block in an optimal solution is given by $t_{min} + max\left\{p_{max}^A, p_{max}^B\right\}$.

Based on Property 4, we claim that in order to find an optimal schedule, our proposed DP should be repeated for all relevant t-values: $t_{min} \leq t \leq t_{min} + \max\left\{p_{max}^A, p_{max}^B\right\}$. Due to Property 3 we sort the A-jobs in a non-decreasing order of p_j^A/w_j^A. For this sorted sequence, we calculate the partial sums: $P_j^A = \sum_{i=1}^{j} p_j^A$, $j = 1, \ldots, n^A$.

For a given block of the B-jobs in the interval $[t, t + P^B]$, we introduce a DP consisting of the following state variables:

$j-$ the number of $A-$jobs already scheduled;

$e-$ the total load (i.e., total processing time) of the A-jobs scheduled already after the B-block. (Note that the completion time of this subset of A jobs is exactly D, and its starting time is $D - e$. It follows that the remaining A-jobs, which are processed continuously, are completed at time t and start at time $t - (P_j^A - e)$; see Fig. 1.)

Let $f_t(j, e)$ denote the optimal total weighted earliness of the jobs $j+1, j+2, \ldots, n^A$ (of Agent A), given a total load e of the A-jobs scheduled after the B-block that starts at time t.

At each stage, the DP considers two options (for given t, j and e): (i) schedule the next A-job as late as possible after the B-block (i.e., to be completed at time $D - e$), or (ii) schedule the next A-job as late as possible prior to the B-block (i.e., to be completed at time $t-(P_j^A - e)$). Note that for a particular t-value, it is possible that both options (i) and (ii) are feasible, that only option (i) is feasible, that only option (ii) is feasible, or that both options are infeasible.

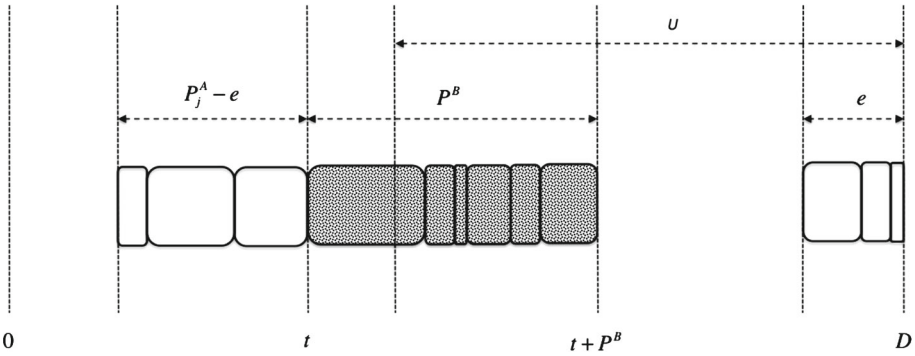

Fig. 1 The state variables and the relevant measures of the DP for minimizing total weighted earliness of the jobs of Agent A, given an upper bound (U) on the maximum earliness of the jobs of Agent B

The recursion is given in the following:

$$f_t(j,e) = \left\{ \begin{array}{ll} \left\{ \begin{array}{l} ew_{j+1}^A + f_t\left(j+1, e+p_{j+1}^A\right) \\ \infty \end{array} \right\} & if\ t + P^B + p_{j+1}^A + e \le D \\ & otherwise \\ \left\{ \begin{array}{l} \left(D - t + \left(P_j^A - e\right)\right) w_{j+1}^A + f_t(j+1,e) \\ \infty \end{array} \right\} & if\ P_{j+1}^A - e \le t \\ & otherwise \end{array} \right\}$$

The boundary conditions are:

$$f_t\left(n^A, e\right) = 0, \quad e = 0, 1, \ldots, D - (t + P^B).$$

The solution is given by: $f_t(0,0)$.
As mentioned above, the DP needs to be repeated for all relevant t-values ($t_{min} \le t \le t_{min} + max\left\{p_{max}^A, p_{max}^B\right\}$), and the global optimum is obtained by:

$$min\left\{f_t, \quad t_{min} \le t \le t_{min} + max\left\{p_{max}^A, p_{max}^B\right\}\right\}.$$

Theorem 1 *The DP finds the optimal solution in* $O\left(\left(n^A\right)^2 p_{max}^A max\left\{p_{max}^A, p_{max}^B\right\}\right)$ *time.*

Proof Each execution of the DP (for a given t-value) requires $O\left(n^A P^A\right) = O\left(\left(n^A\right)^2 p_{max}^A\right)$, since j is bounded by n and e is bounded by P^A. The DP is repeated no more than $max\left\{p_{max}^A, p_{max}^B\right\}$ times. It follows that the total running time is: $O\left(\left(n^A\right)^2 p_{max}^A max\left\{p_{max}^A, p_{max}^B\right\}\right)$. □

We refer the reader to "Appendix 2" for a numerical example (Example 1) demonstrating a solution of a problem consisting of 7 A-jobs and 7 B-jobs.

Numerical Tests: We performed a numerical study in order to evaluate the performance of the proposed DP. Specifically, we measured the running time and storage requirement as a function of the number of the (A) jobs, and the maximum processing time. We considered $n^A = n^B = 25, 50, 75, 100, 125, 150, 175, 200$. The job processing times of both agents A and B were generated uniformly in the interval [1, 100]. (Note that, in fact, for Agent B only

Table 1 Average and worst case running times and memory usage of the DP algorithm as a function of the number of (Agent A) jobs

n^A	Average running time (s)	Worst case running time (s)	Memory usage (MB)
25	0.58	0.62	51
50	2.29	2.73	198
75	4.88	5.23	440
100	9.00	9.81	778
125	13.75	15.93	1211
150	19.68	20.90	1739
175	25.31	26.61	2363
200	33.54	36.42	3082

the total job processing time and the maximum processing time are relevant pieces of data.) The job weights (of Agent A) were generated uniformly in the interval [1, 100]. After the processing times were generated, the total loads (P^A, P^B and D) were computed, and the upper bound on the maximum earliness of Agent B was generated uniformly in the interval $[P^B - p^B_{max}, D - p^B_{max}]$.

For each n^A value, 20 instances were generated and solved, and the running time was measured. Table 1 contains the average and the worst case running times, and the RAM (Random Access Memory) usage for each problem set. The DP was programmed in C, and executed on a Macintosh with a 2.8 GHz Intel Core i7 processor.

Table 1 indicates that the DP solves relatively large problems in very reasonable time. Specifically, solving problems of 200 jobs of Agent A and 200 jobs of Agent B required 33.5 s on average, and the worst case running time did not exceed 36.4 s. In terms of storage requirements, solving these instances required 3.01 GB.

Comment: Note that the entire Pareto optimal set is obtained in pseudo-polynomial time. The DP should be repeated for all possible U values. Recall that $P^B - p^B_{max} \leq U < D - p^B_{max}$. It follows that the DP should be performed $O(P^A)$ times.

4 A heuristic

The above numerical study verifies that the proposed DP appears to perform well for instances of up to 200 jobs. For larger instances, the DP becomes impractical, and an efficient heuristic seems to be necessary. Mor and Mosheiov (2010) proposed an $O(nlogn)$ heuristic for a more general case of multi-agents, based on assigning the jobs of the B-agents as early as possible, and then scheduling the A jobs from D backwards in a non-decreasing order of p^A_j/w^A_j. We adapted this heuristic for the special case of two agents: we first scheduled the B-block as early as possible (with the largest B-job scheduled first in the block), subject to the upper bound on their maximum earliness. Then we scheduled the A jobs (which are sorted in a non-decreasing order of p^A_j/w^A_j) from D backwards. If a positive time interval still remains after the B-block, the starting time of the B-block is delayed, such that this interval is closed. Then, the remaining A jobs are scheduled backwards until time zero. The formal algorithm (denoted *Heuristic* 1) is provided in the following:

Heuristic 1 $(1/d_j = D\ (deadline)/\sum_{j=1}^{n^A} w_j^A E_j^A : E_{max}^B)$:
Input: $p_j^A, w_j^A,\ j = 1, \ldots, n^A,\ \ p_j^B,\ j = 1, \ldots, n^B,\ D, U.$

Step 1: Initialization.
Sort the A-jobs in a non-decreasing order of p_j^A/w_j^A.
Set the first B-job to be the one with largest processing time (p_{max}^B).
Let $Bblock = 0$; /* A binary variable indicating whether the B-block was assigned */
Let $cost = 0$; /* Total weighted earliness of Agent A */

Step 2: Assign A-jobs and B-jobs.
For $j = 1$ to n^A
If $Bblock = 0$ /* The B-Block was not assigned */
/* Check whether assigning the next A-job violates the upper bound constraint on the maximum earliness of Agent B */
If $(e + p_j^A + P^B - p_{max}^B) \leq U$
$cost = cost + e \times w_j^A$; /* Assign the next A-job */
$e = e + p_j^A$;
else
$e = e + P^B$; /* Assign the B-block */
$Bblock = 1$;
else /* The B-Block was assigned */
$cost = cost + e \times w_j^A$;
$e = e + p_j^A$;
Return $cost$.

Theorem 2 *Heuristic 1 runs in $O(n\,logn)$ time.*

Proof Heuristic 1 consists of n iterations, each one is performed in constant time. Hence, the total running time is $O(n\,logn)$ (due to the initial sorting procedure). □

Since the optimality gaps in some test cases were not sufficiently small, we propose in the following a simple and fast improvement procedure. When scheduling the A-jobs backwards, we reach the point where the current A-job in the list cannot be assigned to the remaining interval after the B block (the interval is smaller than the job's processing time). We now look for the next A job in the list, which is sufficiently small to be assigned to this interval. If such a job exists, it is scheduled in the interval. If a positive time interval still remains after the B block, then, as above, the starting time of the B block is delayed, such that this interval is closed. Then, the remaining A jobs are scheduled backwards until time zero. It should be emphasized that although in most cases this additional procedure improves the results, for some instances the value of the weighted earliness may increase. We thus, perform this additional procedure, and clearly choose the new schedule only in case of improvement. The formal algorithm (denoted *Improved Heuristic* 1) is provided in the following:

Improved Heuristic 1 ($1/d_j = D\ (deadline)/\sum_{j=1}^{n^A} w_j^A E_j^A : E_{max}^B$):

Input: $p_j^A, w_j^A,\ j = 1, \ldots, n^A,\ \ p_j^B,\ j = 1, \ldots, n^B,\ D, U.$

Step 1: Initialization.

Sort the A-jobs in a non-decreasing order of p_j^A/w_j^A.

Set the first B-job to be the one with largest processing time (p_{max}^B).

Let $Bblock = 0$; /* A binary variable indicating whether the B-block was assigned */

Let $swapindex = 0$; /* A binary variable indicating the index of the swapped job, 0 means not swap */

Let $cost = 0$; /* Total weighted earliness of Agent A */

Step 2: Assign A-jobs and B-jobs.

For $j = 1$ to n^A

If $Bblock = 0$ /* The B-Block was not assigned */

/* Check whether assigning the next A-job violates the upper bound constraint on the maximum earliness of Agent B */

If $(e + p_j^A + P^B - p_{max}^B) \le U$

$cost = cost + e \times w_j^A$; /* Assign the next A-job */

$e = e + p_j^A$;

else

/* Define $space$ to be the remaining space after the B-block */

$space = U - e - (P^B - p_{max}^B)$;

/* Check whether a sufficiently small A-job ($\le space$) exists*/

for $i = j \to n^A$

If $p_i^A \le space$

$swapIndex = i$;

/* Assign job i after the B-block */

$cost = cost + e \times w_{swapIndex}^A$;

$e = e + p_{swapIndex}^A$;

exit loop;

$e = e + P^B$; /* Assign the B-block */

$Bblock = 1$;

else

if $swapIndex \ne j$ /* If $swapIndex = j$ job j has been already assigned;

continue to the next job */

$cost = cost + e \times w_j^A$;

$e = e + p_j^A$;

Return $cost$.

Table 2 Optimality gaps obtained by *Heuristic* 1 and *Improved Heuristic* 1

n^A	*Heuristic* 1			*Improved Heuristic* 1		
	#opt	Av. opt gap	Worst case	#opt	Av. opt gap	Worst case
25	11	1.021	1.090	11	1.009	1.049
50	3	1.023	1.077	8	1.007	1.041
75	1	1.017	1.044	4	1.004	1.020
100	4	1.013	1.034	5	1.005	1.022
125	3	1.009	1.021	3	1.003	1.010
150	0	1.008	1.024	0	1.002	1.004
175	0	1.007	1.019	0	1.002	1.007
200	0	1.006	1.013	0	1.002	1.012

Theorem 3 *Improved Heuristic 1 runs in* $O(n\,logn)$ *time.*

Proof The additional procedure of moving a single job to be processed after the B-block clearly does not change the running time, which remains $O(n\,logn)$. □

Numerical Tests: We tested both *Heuristic* 1 and *Improved Heuristic* 1. We compared the heuristics' results with those obtained by the DP. Again, we considered $n^A = n^B = 25, 50, 75, 100, 125, 150, 175, 200$. As above, the job processing times of both agents A and B and the job weights of Agent A were generated uniformly from the interval $[1, 100]$. Given the job processing times and the total loads (P^A, P^B and D), the upper bound (U) on the maximum earliness of Agent B was generated uniformly in the interval $[P^B - p^B_{max}, D - p^B_{max}]$.

For a given n^A value, 20 problems were generated. Each problem was solved first by *Heuristic* 1 and by the dynamic programming algorithm. The optimality gap was calculated. Then, *Improved Heuristic* 1 was performed, and the new optimality gap was calculated. For each problem set, the number of optimal schedules obtained by the heuristics, and the average and the worst case optimality gaps were calculated. The results are reported in Table 2.

Table 2 indicates that *Improved Heuristic*1 performs extremely well. For example, the average optimality gap for 100-job problems (which were solved in less than 1 ms) was 1.005. Given these results, it seems clear that *Improved Heuristic*1 is a practical procedure, which is appropriate for real-life settings.

5 A general multi-agent setting

In the following we focus on the more general setting of multi agents. We define A-type agents as agents who need to minimize the total weighted earliness of their jobs. Similarly, each of the B-type agents has an upper bound on the maximum earliness of his jobs.

First, consider a setting of several A-type agents and a single B-type agent. This setting is easily shown to be reduced to a two-agent setting, since all the jobs of the A-type agents can be aggregated into one set of a single (A) agent. Even the more general case, where different A-agents have different weights, can be reduced to a two-agent problem. Consider the case of two A-agents with weights θ_1 and θ_2, respectively. The objective function is minimum $\theta_1 \left(\sum_{j=1}^{n^{A_1}} w_j^{A_1} E_j^{A_1}\right) + \theta_2 \left(\sum_{j=1}^{n^{A_2}} w_j^{A_2} E_j^{A_2}\right)$. This function can be replaced by: $\sum_{j=1}^{n^A} w_j^A E_j^A$ where $w_j^A = \theta_1 w_j^{A_1}$ for $j \in J^{A_1}$ and $w_j^A = \theta_2 w_j^{A_2}$ for $j \in J^{A_2}$.

Consider now a setting of m B-type agents and a number of A-type agents. First, as explained above, the A-type agents are aggregated into a single agent. The problem is reduced to that of m B-type agents and a single A-type agent. If m is not part of the input (i.e., is not a given constant) the problem is known to be NP-hard in the strong sense (see Mor and Mosheiov 2010). We thus consider in the following the case of a *given* $m(\geq 2)$ value. We show that the pseudo-polynomial DP introduced in Sect. 3 can be extended to a setting of two or more B-agents, implying that this more general problem remains NP-hard in the ordinary sense.

For ease of exposition we focus in the following on the case of two B-agents ($m = 2$). [For the case of general m ($m \geq 2$), we refer the reader to "Appendix 3".] We extend the notation accordingly. Agent B_i has a set J^{B_i} of n^{B_i} jobs, $i = 1, 2$. The processing time of job j in the set J^{B_i} is denoted by $p_j^{B_i}$, $j = 1, \ldots, n^{B_i}$, $i = 1, 2$. Let $P^{B_i} = \sum_{j=1}^{n^{B_i}} p_j^{B_i}$ denote the total processing time of the jobs of Agent B_i, $i = 1, 2$. Let $p_{max}^{B_i} = max\left\{p_j^{B_i}, j = 1, \ldots, n^{B_i}\right\}$ denote the maximal processing time of the jobs of Agent B_i, $i = 1, 2$. The common deadline for all the jobs of all agents is $D = P^A + \sum_{i=1}^{2} P^{B_i}$. Let U_i be the upper bound on the maximum earliness of B_i agent , $i = 1, 2$. Let t_i denote the starting time of block B_i, $i = 1, 2$.

First, we claim that Properties 1, 2 and 3 are easily extended to this new setting as follows:

Property 1 ($m = 2$): An optimal schedule exists in which all the jobs of agents B_1 and B_2 are scheduled (continuously) in a single block, respectively.

Property 2 ($m = 2$): An optimal schedule exists in which for both agents B_1 and B_2, the largest job is scheduled first in his block, respectively.

Property 3 ($m = 2$): An optimal schedule exists in which the A-jobs are scheduled in (at most) 3 blocks, and the jobs in each block are sequenced in a non-increasing order of p_j^A/w_j^A.

It should be noted that according to Property 3 ($m = 2$) the number of B-blocks may reduce to one. This will happen when the two B-blocks are processed continuously with no A-jobs among them.

Without loss of generality we assume $t_1 \leq t_2$ (the B-blocks are renumbered according to their processing order). Thus, a lower bound on t_1 is given by: $t_{min}^{B_1} = \left\{D - U_1 - p_{max}^{B_1}\right\}$. An upper bound (to avoid a trivial solution, see Property 4) is $t_{min}^{B_1} + \left\{p_{max}^{A}, p_{max}^{B_1}\right\}$. For any realization of t_1, a lower bound on t_2 is $t_{min}^{B_2} = \max\left\{D - U_2 - p_{max}^{B_2}, t_1 + P_1^{B_1}\right\}$. Thus, the upper bound on t_2 is $t_{min}^{B_2} + \max\left\{p_{max}^{A}, p_{max}^{B_2}\right\}$. Formally,

Property 4 ($m = 2$): An optimal schedule exists such that the starting times of the B_i-blocks are bounded by: $t_{min}^{B_1} \leq t_1 \leq t_{min}^{B_1} + \max\left\{p_{max}^{A}, p_{max}^{B_1}\right\}$, and $t_{min}^{B_2} \leq t_2 \leq t_{min}^{B_2} + \max\left\{p_{max}^{A}, p_{max}^{B_2}\right\}$.

As mentioned, the A-jobs are scheduled in (at most) 3 blocks. For convenience we denote them by A_1 (the earliest), A_2 and A_3 (the latest). Following the DP introduced for two agents, our extended DP assigns A-jobs in each block from the end towards the beginning. Hence we initially sort the jobs in a non-decreasing order of p_j^A/w_j^A (Property 3). The extended DP requires the following state variables:

j The number of $A-$jobs already scheduled;
e_1 The total processing time of the A-jobs scheduled after block B_1 and prior to block B_2.
e_2 The total processing time of the A-jobs scheduled after block B_2.

Note that $P_j^A - e_1 - e_2$ is the total processing time of the A-jobs scheduled prior to block B_1. All three A-blocks are scheduled in non-increasing order of job processing times. The first block of the A-jobs is processed in the interval $[t_1 - \left(P_j^A - e_1 - e_2\right), t_1]$. The second block of the A-jobs is processed in the interval $[t_2 - e_1, t_2]$. The third block of the A-jobs is processed in the interval $[D - e_2, D]$.

Let $f_{t_1,t_2}(j, e_1, e_2)$ denote the optimal total weighted earliness of the jobs $j+1, j+2, \ldots, n^A$ (of Agent A), given a total load of e_i of the A-jobs scheduled in block A_i (after the B_i-block that starts at time t_i), $i = 1, 2$.

At each stage, the DP considers 3 options: (i) schedule the next A-job as late as possible prior to the first A block; or (ii) schedule the next A-job as late as possible prior to the second A block; or (iii) schedule the next A-job as late as possible prior to the third A block. The recursion is given in the following:

$$f_{t_1,t_2}(j, e_1, e_2) = min \left\{ \begin{array}{ll} \left\{ \begin{array}{c} \left(D - t_1 + \left(P_j^A - e_1 - e_2\right)\right) w_{j+1}^A + f_{t_1,t_2}(j+1, e_1, e_2) \\ \infty \end{array} \right\} & \begin{array}{l} if\ P_{j+1}^A - e_1 - e_2 \le t_1 \\ otherwise \end{array} \\ \left\{ \begin{array}{c} (D - t_2 + e_1) w_{j+1}^A + f_{t_1,t_2}\left(j+1, e_1 + p_{j+1}^A, e_2\right) \\ \infty \end{array} \right\} & \begin{array}{l} if\ t_1 + P^{B_1} + e_1 + p_{j+1}^A \le t_2 \\ otherwise \end{array} \\ \left\{ \begin{array}{c} e_2 w_{j+1}^A + f_{t_1,t_2}\left(j+1, e_1, e_2 + p_{j+1}^A\right) \\ \infty \end{array} \right\} & \begin{array}{l} if\ t_2 + P^{B_2} + e_2 + p_{j+1}^A \le D \\ otherwise \end{array} \end{array} \right\}$$

The boundary conditions are:

$$f_{t_1,t_2} = f_{t_1,t_2}\left(n^A, e_1, e_2\right) = 0, \quad e_1,\ e_2 = 0, 1, \ldots, P^A, \quad e_1 + e_2 \le P^A.$$

The solution is given by: $f_{t_1,t_2}(0, 0, 0)$.
The DP needs to be executed for all relevant combinations of t_1 and t_2, leading to the following total running time:

Theorem 4 *The DP finds the optimal solution in* $O\left(\left(n^A\right)^3 \left(p_{max}^A\right)^2 \times \left(max\left\{p_{max}^A, p_{max}^{B_1}\right\}\right) \times \left(max\left\{p_{max}^A, p_{max}^{B_2}\right\}\right)\right)$ *time.*

Proof Each execution of the DP (for given t_1 and t_2) requires $O\left(n^A \left(P^A\right)^2\right) = O\left(\left(n^A\right)^3 \left(p_{max}^A\right)^2\right)$. The DP is repeated no more than $\left(max\left\{p_{max}^A, p_{max}^{B_1}\right\}\right) \times \left(max\left\{p_{max}^A, p_{max}^{B_2}\right\}\right)$ times. Therefore, the total running time is:

$$O\left(\left(n^A\right)^3 \left(p_{max}^A\right)^2 \times \left(max\left\{p_{max}^A, p_{max}^{B_1}\right\}\right) \times \left(max\left\{p_{max}^A, p_{max}^{B_2}\right\}\right)\right).$$

□

As mentioned above, we refer the reader to "Appendix 3" for generalization of the DP to the setting of any given m ($m \ge 2$) B-Agents.

6 A dynamic programming algorithm for $1/d_j = D(deadline)/E^B_{max} : \sum_{j=1}^{n^A} w^A_j E^A_j$

In this section we focus on the inverse problem of minimizing the maximum earliness of Agent B, subject to an upper bound on the total weighted earliness of Agent A. This problem is NP-hard since the original problem was proved to be NP-hard. (Recall that the recognition version of both problems is identical:
$1/d_j = D\,(deadline)/E^B_{max} \leq U_1 : \sum_{j=1}^{n^A} w^A_j E^A_j \leq U_2$ for two given upper bounds U_1 and U_2.) We thus introduce a dynamic programming algorithm, which is an adaptation of the DP introduced in Sect. 3. First, it is easily verified that *Property* 1 (a single B-block), *Property* 2 (the largest B-job is scheduled first), and *Property* 3 (A-jobs are scheduled in a non-increasing order of p^A_j/w^A_j), continue to hold. The new Property 4 (referring to the possible starting time t of the B-block) is the following:

Property 4′ The starting time t of the B-block is bounded to be within the interval $[0,\ D-P^B]$.

Proof Trivial. Due to the common deadline D, the B-block cannot start after $D - P^B$.

Given these properties, we propose in the following a dynamic programming algorithm which is a minor modification of the one introduced in Sect. 3. For a given t value, we use the same state variables: j–the number of $A-$jobs already scheduled; e–the total processing time of the A-jobs scheduled after to the B-block. As above, $f_t\,(j, e)$ denotes the optimal total weighted earliness of the jobs $j+1, j+2, \ldots, n^A$ (of Agent A), given a total load e of the A-jobs scheduled after the B-block that starts at time t. The recursion and the boundary conditions remain valid. The optimal total weighted earliness (for a given t) is given by $f_t\,(0, 0)$. After performing the DP for a given t-value, we check whether the solution is feasible: if $f_t\,(0, 0) \leq U_2$ then the solution is feasible and is registered. The DP is repeated for all relevant t values (according to *Property* 4′), and the optimal t value is given by: $t^{opt} = argmax\{f_t\,(0, 0) \leq U_2; t = 0, \ldots, D - P^B\}$. The optimal maximal earliness of Agent B is given by $E^B_{max} = D - t^{opt} - p^B_{max}$. (Note that $f_t\,(0, 0)$ is not necessarily monotone in t, implying that a binary search is not sufficient, and all the t values in the interval $[0,\ D - P^B]$ need to be checked.) □

Theorem 5 *The DP finds the optimal solution in* $O\left(\left(n^A\right)^3\left(p^A_{max}\right)^2\right)$ *time.*

Proof The running time of the DP for a given t-value remains $O\left(n^A P^A\right) = O\left(\left(n^A\right)^2 p^A_{max}\right)$. The DP is repeated $D - P^B = P^A$ times, implying that the total running time is: $O\left(\left(n^A\right)^2 p^A_{max} P^A\right) = O\left(\left(n^A\right)^3\left(p^A_{max}\right)^2\right)$. □

Heuristic and Lower Bound: While the pseudo-polynomial DP proves that
$1/d_j = D(deadline)/E^B_{max} : \sum_{j=1}^{n^A} w^A_j E^A_j$ is NP-hard in the ordinary sense, it appears to be not very practical. A 25-job problem (where job processing times are generated uniformly from the interval [1, 100]) is solved in 3.7 s on average, and the worst case requires 4.4 s. A 50-job problem requires 29.6 s on average, and the worst case time becomes 36.8 s. We thus introduce in the following a simple heuristic and a lower bound.

The proposed heuristic is of a greedy type. We start with an initial solution in which all the jobs of Agent A are scheduled continuously (in a non-increasing order of p^A_j/w^A_j), in the interval $[0,\ P^A]$. The B-block is scheduled after the A jobs, in the interval $[P^A,\ D]$. The total weighted earliness of Agent A is calculated, and if it exceeds the upper bound (i.e., if

$\sum_{j=1}^{n^A} w_j^A E_j^A > U$), the last scheduled A job (immediately preceding the B-block) and the B-block are replaced. It is clear that the new schedule has a smaller $\sum_{j=1}^{n^A} w_j^A E_j^A$ value, (but larger E_{max}^B value). The new weighted earliness of Agent A is calculated, and if it becomes feasible we stop. Otherwise, another replacement is performed. The procedure continues until a feasible schedule is reached. The formal algorithm (denoted *Heuristic* 2) is the following:

Heuristic 2 $(1/d_j = D\ (deadline)/E_{max}^B\colon\ \sum_{j=1}^{n^A} w_j^A E_j^A)$
Input: $p_j^A, w_j^A,\ j = 1, \ldots, n^A,\quad p_j^B,\ j = 1, \ldots, n^B,\ D, U.$

Step 1: Initialization.
Sort the A-jobs in a non-increasing order of p_j^A/w_j^A. Schedule them in the interval $[0, P^A]$.
Schedule the B-jobs in the interval $[P^A, D]$. Set the first B-job to be the largest one.

Step 2: Find a feasible schedule.
Let $cost = \sum_{j=1}^{n^A} w_j^A E_j^A$; /* Initial weighted earliness of Agent A */
Let $E_{max}^B = D - P^B - p_{max}^B$; /* Initial maximum earliness of Agent B */
For $j = n^A$ to 1
If $cost > U$ /* Infeasible scheduled */
/* Replace the next A-job (j) with the B-block */
$cost = cost - w_j^A P^B$;
$E_{max}^B = E_{max}^B + p_j^A$;
else
return E_{max}^B;
Return ∞; /* No feasible solution was found */

Theorem 6 *Heuristic 2 runs in $O(n\,logn)$ time.*

Proof Step 2 is performed at most n times, and each iteration requires constant time, leading to an $O(n)$ effort. Hence, the total running time is $O(n\,logn)$ (due to the initial sorting procedure). □

In order to evaluate the heuristic's performance a tight lower bound is required. Such a lower bound is trivially obtained by considering the initial sequence of the A-jobs (sorted in a non-increasing order of p_j^A/w_j^A), and assigning the B-block as late as possible, while allowing *job preemption*. Formally, we start with the B-block assigned to the interval $[P^A, D]$, and at each iteration it is moved to start a single unit of time earlier. The lower bound is obtained when a (preempted) schedule is reached with total weighted earliness of Agent A, which is not larger than the upper bound U.
We refer the reader to "Appendix 2" for a numerical example (Example 2). In this example all relevant values are calculated: the optimum obtained by the DP, the result of Heuristic 2 and the value of the lower bound.
Numerical tests: we tested the performance of *Heuristic* 2 against the above proposed lower bound. As above, we considered $n^A = n^B = 25, 50, 75, 100, 125, 150, 175, 200$, the job processing times of both agents A and B and the job weights of Agent A were generated uniformly in the interval [1, 100], and the upper bound (U) on the maximum earliness of Agent B was generated uniformly in the interval $[P^B - p_{max}^B,\ D - p_{max}^B]$.

Again, for a given n^A value, 20 problems were generated. Each problem was solved by *Heuristic* 2 and the lower bound was calculated. The optimality gap (E_{max}^B/LB) was

Table 3 Optimality gaps obtained by *Heuristic* 2

n^A	Average optimality gap	Worst case
25	1.0200	1.0383
50	1.0107	1.0189
75	1.0058	1.0102
100	1.0047	1.0074
125	1.0041	1.0071
150	1.0028	1.0060
175	1.0027	1.0043
200	1.0020	1.0039

computed. For each problem set, the average and the worst case optimality gaps are reported in Table 3. The results verify that both *Heuristic* 2 and the lower bound are extremely accurate. (It should be noted that the heuristic complexity is $O(nlogn)$ time only, and the actual average running time required for solving a 200-job problem, for example, was 0.003 ms.)

7 Conclusion

We studied a single machine two-agent scheduling problem where the objective function is minimum total weighted earliness of the first agent subject to an upper bound on the maximum earliness of the jobs of the second agent. In a recent paper, Mor and Mosheiov (2010) studied this problem and proved NP-hardness. In this paper we introduced a pseudo-polynomial dynamic programming algorithm, proving NP-hardness in the ordinary sense. We showed that the DP performs well even for relatively large instances. The DP was extended to a setting of any (given) number of agents, proving that this more general setting remains NP-hard in the ordinary sense. A simple heuristic was also introduced and tested numerically. We also studied the inverse problem of minimizing the maximum earliness of one agent, subject to an upper bound on the maximum weighted earliness of the other agent. We introduced a pseudo-polynomial dynamic programming for this (NP-hard) problem, a greedy-type heuristic and a tight lower bound based on the preemptive schedule. Our numerical tests led to extremely small optimality gaps.

Extending the problems studied in this paper to more general machine settings (starting with parallel identical machines) is a potential topic for future research. Also, considering a different earliness measure for Agent B (total earliness, total weighted earliness, number of early jobs) appears to be a challenging line of research.

Acknowledgements This research was supported by the Israel Science Foundation (grant No.1286/14). The third author was supported in part by the Recanati Fund of The School of Business Administration, and Charles I. Rosen Chair of Management, The Hebrew University of Jerusalem, Israel.

Appendix 1: Proofs of Properties 1–4

Proof of Property 1 Consider an optimal schedule, say q, in which an A-job is scheduled between B-jobs. By moving this A job to be scheduled after all the B-jobs, we obtain a new feasible schedule q', with a strictly smaller total weighted earliness of the A-jobs. This contradicts the optimality of schedule q. □

Proof of Property 2 Consider an optimal schedule q consisting of a single block of the B-jobs (Property 1), in which the first B-job (say k) is not the largest, and the largest B-job (say l) is scheduled later. Assume that this block starts at time t. Clearly, the maximum earliness of the B-jobs is given by $D-(t+p_k)$. Create a new schedule q' by moving job l to be first and start at t. q' is clearly feasible (the maximum earliness becomes smaller: $D-(t+p_l)$), with no impact on the total weighted earliness of the A-jobs, implying that q' is optimal as well. □

Proof of Property 3 The two-block claim follows Property 1. The order of the jobs within each block is easily proved by a standard pair-wise interchange argument. □

Proof of Property 4 Consider an optimal schedule q with $t > t_{min} + \max\left\{p^A_{max}, p^B_{max}\right\}$. Create a new schedule q' by replacing the last A-job scheduled prior to the B-block with the B-block. It is clear that q' is feasible since the new t value is larger than the original (and in particular not smaller than t_{min}). Moreover, the total weighted earliness of the A-jobs in q' is strictly smaller than that of q, contradicting the optimality of q. □

Appendix 2: numerical examples

Example 1 (for problem $1/d_j = D(deadline)/\sum_{j=1}^{n^A} w^A_j E^A_j : E^B_{max}$):

Consider a 2-agent problem, where each agent needs to process 7 jobs. The job processing times of Agent A are: $p^A_1=1,\ p^A_2=2,\ p^A_3=7,\ p^A_4=8,\ p^A_5=4,\ p^A_6=5,\ p^A_7=6$. The weights of the jobs of Agent A are: $w^A_1=8,\ w^A_2=2,\ w^A_3=7,\ w^A_4=1,\ w^A_5=3,\ w^A_6=4,\ w^A_7=2$. The job processing times of Agent B are: $p^B_1=8,\ p^B_2=2,\ p^B_3=1,\ p^B_4=3,\ p^B_5=7,\ p^B_6=2,\ p^B_7=4$. The upper bound on the maximum earliness of Agent B is $U=33$. It follows that $P^A=33,\ P^B=27$, and $D=P^A+P^B=60$. The smallest possible starting time of the B-block is $t_{min}=D-U-p^B_{max}=19$. The largest possible starting time of the B-block is $t_{min}+max\left\{p^A_{max}, p^B_{max}\right\}=27$. The optimal solution obtained by the DP given in Sect. 3 (see Fig. 2) consists of the following job sequence: Job 4 of Agent A is first (starts at zero and is completed at 8); Job 7 of Agent A (completed at 14); Job 6 of Agent A (completed at 19); The block of all the B-jobs (in the interval [19, 46]); Job 5 of Agent A (completed at 50); Job 3 of Agent A (completed at 57); Job 2 of Agent A (completed at 59); Job 1 of Agent A (completed at $D=60$). The total cost (weighted earliness of Agent A) is 361.

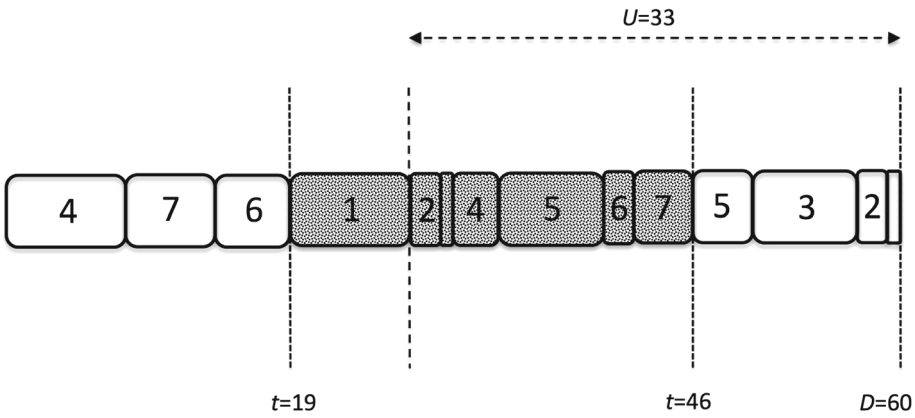

Fig. 2 The optimal solution of the two-agent problem solved in Example 1

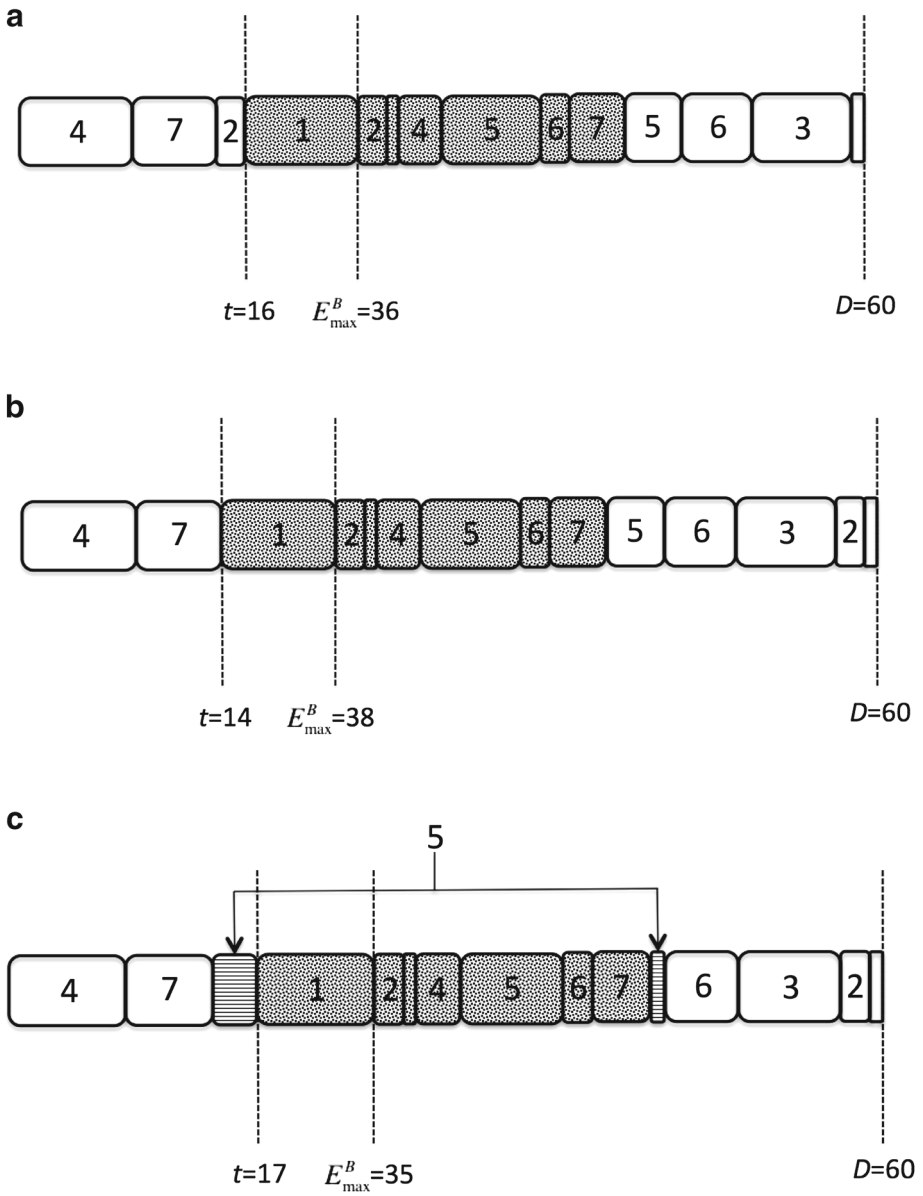

Fig. 3 **a** The optimal solution of the two-agent problem solved in Example 2, **b** the schedule obtained by Heuristic 2 in Example 2, **c** the lower bound obtained in Example 2 based on a preemptive schedule

Example 2 (for problem $1/d_j = D\,(deadline)/E^B_{max} : \sum_{j=1}^{n^A} w^A_j E^A_j$):

We use the same data of Example 1 (processing times of both agents, and weights of Agent A). As above, $P^A = 33$, $P^B = 27$, $D = P^A + P^B = 60$. The upper bound on the total weighted earliness of Agent A is $U = 310$. The interval of a feasible t-values is $[0, 33]$. The *optimal solution* obtained by the DP given in Sect. 6 (see Fig. 3a) consists of the following

job sequence: Job 4, Job 7, Job 2, the B-block, Job 5, Job 6, Job 3, Job 1. $t^{opt} = 16$. The optimal maximal earliness of Agent B is $E^B_{max} = D - t^{opt} - p^B_{max} = 60 - 16 - 8 = 36$.

We now solve the problem using *Heuristic* 2. The schedule obtained by the heuristic (see Fig. 3b) consists of the following job sequence: Job 4, Job 7, the B-block, Job 5, Job 6, Job 3, Job 2, Job 1. $t^{heur} = 14$. Thus, the maximal earliness of Agent B obtained by Heuristic 2 is $E^B_{max} = D - p^B_{max} = 60 - 14 - 8 = 38$. Note that the total weighted earliness of Agent A is 252 (smaller than $U = 310$).

Finally, we calculate the *lower bound* obtained by the following preempted schedule (see Fig. 3c): Job 4, Job 7, Job 5 (starts at time 14 and is preempted at time 17 by the B-block), the B-block (which starts at time 17 and is completed at time 44), the last unit of time of Job 5, Job 6, Job 3, Job 2, Job 1. $t^{LB} = 17$. The lower bound on the maximal earliness of Agent B is $E^B_{max} = D - t^{LB} - p^B_{max} = 60 - 17 - 8 = 35$.

In summary: the optimal E^B_{max} in Example 2 (obtained by the DP) is 36, the heuristic value is 38, and the lower bound is 35.

Appendix 3: a dynamic programming for the case of m B-agents

We now assume that each Agent B_i has a set J^{B_i} of n^{B_i} jobs, $i = 1, \ldots, m$. The above definitions of $p^{B_i}_j$, P^{B_i}, $p^{B_i}_{max}$, D, U_i and t_i remain unchanged, $i = 1, \ldots, m$. Clearly, Properties 1–3 remain valid. Similar to the case of $m = 2$, the number of B-blocks may reduce when two (or more) B-blocks are processed continuously with no A-jobs among them. Again, without loss of generality we assume $t_1 \le t_2 \le \cdots \le t_m$. A lower bound on t_i is given by: $t^{B_i}_{min} = D - U_i - p^{B_i}_{max}$. An upper bound is $t^{B_i}_{min} + \max\left\{p^A_{max}, p^{B_i}_{max}\right\}$, $i = 1, \ldots, m$. The extension of Property 4 to the setting of m agents is that an optimal schedule exists such that the starting times of the B_i-blocks are bounded by: $t^{B_i}_{min} \le t_i \le t^{B_i}_{min} + \max\left\{p^A_{max}, p^{B_i}_{max}\right\}$, $i = 1, \ldots, m$.

The A-jobs are scheduled in (at most) $m + 1$ blocks. We sort the jobs in a non-decreasing order of p^A_j / w^A_j. The updated state variables are the following:

j The number of $A-$jobs already scheduled;
e_i The total processing time of the A-jobs scheduled after block B_i and prior to block B_{i+1}, $i = 1, \ldots, m-1$
e_m The total processing time of the A-jobs scheduled after block B_m.

$P^A_j - \sum_{i=1}^m e_i$ is the total processing time of the A-jobs scheduled prior to block B_1.
Let $f_{t_1,\ldots,t_m}(j, e_1, \ldots, e_m)$ denote the optimal total weighted earliness of the jobs $j+1, j+2, \ldots, n^A$ (of Agent A), given a total load of e_i of the A-jobs scheduled after the B_i-block, $i = 1, \ldots, m$.
The DP compares $m + 1$ options at each iteration: schedule the next A-job as late as possible after block $B_i, i = 1, \ldots m$, or as late as possible prior to block B_1. The recursion becomes:

$$f_{t_1,\ldots,t_m}(j,e_1,\ldots,e_m)$$

$$= min \left\{ \begin{array}{ll} \left\{ \begin{array}{l} \left(D - t_1 + \left(P_j^A - \sum_{i=1}^m e_i\right)\right) w_{j+1}^A + f_{t_1,\ldots,t_m}(j+1,e_1,\ldots,e_m) \\ \infty \end{array} \right\} & if\ P_{j+1}^A - \sum_{i=1}^m e_i \le t_1 \\ & otherwise \\ \left\{ \begin{array}{l} (D - t_2 + e_1) w_{j+1}^A + f_{t_1,\ldots,t_m}\left(j+1, e_1 + p_{j+1}^A, e_2,\ldots,e_m\right) \\ \infty \end{array} \right\} & if\ t_1 + P^{B_1} + e_1 + p_{j+1}^A \le t_2 \\ & otherwise \\ \left\{ \begin{array}{l} (D - t_3 + e_2) w_{j+1}^A + f_{t_1,\ldots,t_m}\left(j+1, e_1, e_2 + p_{j+1}^A,\ldots,e_m\right) \\ \infty \end{array} \right\} & if\ t_2 + P^{B_2} + e_2 + p_{j+1}^A \le t_3 \\ & otherwise \\ \ldots & \\ \ldots & \\ \ldots & \\ \left\{ \begin{array}{l} e_m w_{j+1}^A + f_{t_1,\ldots,t_m}\left(j+1, e_1,\ldots,e_{m-1}, e_m + p_{j+1}^A\right) \\ \infty \end{array} \right\} & if\ t_m + P^{B_m} + e_m + p_{j+1}^A \le D \\ & otherwise \end{array} \right\}$$

The boundary conditions are:

$$f_{t_1,\ldots,t_m} = f_{t_1,\ldots,t_m}(n^A, e_1,\ldots,e_m) = 0,\ \ e_1,\ldots,e_m = 0,1,\ldots,P^A,\ \ e_1 + \ldots + e_m \le P^A.$$

The solution is given by: $f_{t_1,\ldots,t_m}(0,0,\ldots,0)$.
The DP needs to be executed for all relevant combinations of t_i, which leads to the following:

Theorem 7 *For the general setting of m B-agents, the running time of the DP is:* $O\left(\left(n^A\right)^{m+1}\left(p_{max}^A\right)^m \times \left(max\left\{p_{max}^A, p_{max}^B\right\}\right)^m\right)$ *time, where* $p_{max}^B = \max\left\{p_{max}^{B_i}, i = 1,\ldots m\right\}$

Proof $e_i, i = 1,\ldots,m$ is bounded by P^A, which is bounded by $n^A p_{max}^A$. Therefore, for given $t_1, t_2, \ldots, t_m$, the running time is $O\left(\left(n^A\right)^{m+1}\left(p_{max}^A\right)^m\right)$. The DP is repeated up to $\left(max\left\{p_{max}^A, p_{max}^B\right\}\right)^m$ times. Thus, the total running time is:

$$O\left(\left(n^A\right)^{m+1}\left(p_{max}^A\right)^m \times \left(max\left\{p_{max}^A, p_{max}^B\right\}\right)^m\right).$$

□

References

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

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

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

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

Cheng, S. R. (2014a). Scheduling two-agents with a time-dependent deterioration to minimize the minsum earliness measures. *Asia-Pacific Journal of Operational Research*, *31*, 1–10.

Cheng, S. R. (2014b). Some new problems on two-agent scheduling to minimize the earliness costs. *International Journal of Production Economics*, *156*, 24–30.

Cheng, T. C. E., Chung, Y.-H., Liao, S.-C., & Lee, W.-C. (2013). Two-agent single-machine scheduling with release times to minimize the total weighted completion time. *Computers & Operations Research*, *40*, 353–361.

Donatas, E., & T'kindt, V. (2014). Two-agent scheduling on uniform parallel machines with min–max criteria. *Annals of Operations Research*, *213*, 79–94.

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

Gerstl, E., & Mosheiov, G. (2013). Scheduling problems with two competing agents to minimized weighted earliness–tardiness. *Computers & Operations Research*, *40*, 109–116.
Gerstl, E., & Mosheiov, G. (2014). Single machine just-in-time scheduling problems with two competing agents. *Naval Research Logistics*, *61*, 1–16.
Kellerer, H., & Strusevich, V. A. (2010). Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications. *Algorithmica*, *57*, 769–795.
Li, D.-C., & Hsu, P.-H. (2012). Solving two-agent single-machine scheduling problem considering learning effect. *Computers & Operations Research*, *39*, 1644–1651.
Mor, B., & Mosheiov, G. (2010). Scheduling problems with two competing agents to minimize minmax and minsum earliness measures. *European Journal of Operational Research*, *206*, 540–546.
Mor, B., & Mosheiov, G. (2011). Single machine batch scheduling with two competing agents to minimize total flowtime. *European Journal of Operational Research*, *215*, 524–531.
Oron, D., Shabtay, D., & Steiner, G. (2015). Single machine scheduling with two competing agents and equal job processing times. *European Journal of Operational Research*, *244*, 86–99.
Wan, G., Vakati, S. R., Leung, J. Y.-T., & Pinedo, M. (2010). Scheduling two agents with controllable processing times. *European Journal of Operational Research*, *205*, 528–539.
Yin, Y., Cheng, S. R., & Wu, C. C. (2012). Scheduling problems with two agents and a linear non-increasing deterioration to minimize earliness penalties. *Information Sciences*, *189*, 282–292.
Zuobao, W., & Weng, M. X. (2005). Multiagent scheduling method with earliness and tardiness objectives in flexible job shops. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *35*, 293–301.