J Sched (2016) 19:729–736
DOI 10.1007/s10951-015-0464-7

Scheduling on parallel identical machines with late work criterion: Offline and online cases

Xin Chen[1] · **Malgorzata Sterna**[2] · **Xin Han**[1] · **Jacek Blazewicz**[2]

Published online: 21 December 2015

Abstract In the paper, we consider the problem of scheduling jobs on parallel identical machines with the late work criterion and a common due date, both offline and online cases. Since the late work criterion has not been studied in the online mode so far, the analysis of the online problem is preceded by the analysis of the offline problem, whose complexity status has not been formally stated in the literature yet. Namely, for the offline mode, we prove that the two-machine problem is binary NP-hard, and the general case is unary NP-hard. In the online mode we assume that jobs arrive in the system one by one, i.e., we consider the *online over list* model. We give an algorithm with a competitive ratio being a function of the number of machines, and we prove the optimality of this approach for two identical machines.

Keywords Online and offline scheduling · Identical parallel machines · Late work · NP-hardness · Competitive ratio

✉ Malgorzata Sterna
malgorzata.sterna@cs.put.poznan.pl

Xin Chen
cx.dlut@gmail.com

Xin Han
hanxin@dlut.edu.cn

Jacek Blazewicz
jacek.blazewicz@cs.put.poznan.pl

[1] School of Software Technology, Dalian University of Technology, Tuqiang Street 321, 116620 Dalian, China

[2] Institute of Computing Science, Poznan University of Technology, Piotrowo 2, 60-965 Poznan, Poland

1 Introduction

Time constraints, which can be used to determine feasibility conditions, as well as to evaluate the quality of feasible solutions, play an important role in scheduling problems which can be met in the real world. In the scheduling theory, time constraints might be modeled with due dates or deadlines and the quality of schedules is estimated with reference to these parameters, leading to such criteria as lateness (e.g., McMahon and Florian 1975), tardiness (e.g., Emmons 1969), or the number of tardy jobs (e.g., Moore 1968).

Late work criterion is one of the less explored objective functions based on due dates. It was first proposed by Blazewicz (1984). Soon a number of research groups focused on this performance measure, obtaining a set of interesting results. The single-machine case was considered mainly by Potts and Van Wassenhove (1991), Hochbaum and Shamir (1990), Hariri et al. (1995), Kovalyov et al. (1994), Kethley and Alidaee (2002), and, more recently, by Lin and Hsu (2005), Zhang and Wang (2005), as well as by Ren et al. (2009). The parallel machines environment was studied by Blazewicz (1984), Blazewicz and Finke (1987), and Leung (2004), while the dedicated machines environment was investigated mainly by Blazewicz et al. (2004a, 2005a, 2007) and by Leung (2004), as well as by Lin et al. (2006). The state of the art for late work scheduling was presented by Leung (2004) in the context of imprecise computations and then by Sterna (2011). This latter survey shows that the majority of results obtained for the late work criterion so far concern the single-machine or shop systems, and not too much attention has been paid to the parallel machines environment. Besides, all results presented in the literature concern the offline version of the mentioned scheduling problems, and no paper is devoted to the online case.

In this work, we consider the problem of scheduling jobs on parallel identical machines with the late work criterion and a common due date, both offline and online versions. For the offline case, we prove that the problem for two machines ($P2|d_j = d|Y$) is binary NP-hard, while for an arbitrary number of machines ($P|d_j = d|Y$), it is unary NP-hard. In the online model studied in this paper, jobs appear in the system one by one: when the previous job is scheduled, the next one may arrive. Since there is an input sequence of jobs, this model is called in the literature online "over list." For the online version of the analyzed scheduling problem ($P|d_j = d, online\ over\ list|Y$), we propose an algorithm with competitive ratio $\frac{\sqrt{2m^2-2m+1}-1}{m-1}$, where m is the number of machines. Then, we prove the optimality of this method for two identical machines. More precisely, we show that when $m = 2$, the competitive ratio, equal to $\sqrt{5}-1$, is identical with the lower bound of the problem, so the proposed algorithm is optimal for $P2|d_j = d, online\ over\ list|Y$. Moreover, when $m \to \infty$, the competitive ratio converges to $\sqrt{2}$, which is constant.

The rest of the paper is organized as follows: Section 2 presents the formal definition of the considered problem and provides some information on the related work. Section 3 shows that the offline problem is NP-hard. The online case is investigated in Sect. 4, where an online algorithm with the constant competitive ratio is proposed together with the proof of its optimality for two machines. Some conclusions and future research directions are given in Sect. 5.

2 Problem definition and related work

The problem of scheduling jobs on parallel identical machines with the late work criterion can be defined as follows: There are given n jobs $J = \{J_1, \ldots, J_n\}$ and m identical machines $M = \{M_1, \ldots, M_m\}$. Each job J_j ($j = 1, \ldots, n$) is described by its processing time p_j, due date d_j (representing the preferred completion time for this job), and optionally weight w_j (representing the relative importance of this job). In this paper, we consider a common due date for all the jobs (i.e., $d_j = d$, for $j = 1, \ldots, n$), and we assume that jobs have the same unit weight (i.e., $w_j = 1$, for $j = 1, \ldots, n$). To solve the problem, it is necessary to schedule all jobs (i.e., to assign jobs to machines and to determine their sequence on particular machines), such that all jobs are executed without preemption and each machine executes at most one job at the same time (i.e., for each job J_j its feasible completion time C_j is determined). The schedule should minimize the total late work Y.

The late work for job J_j is equal to the length of the late part of this job (if any), i.e., $Y_j = \min\{p_j, \max\{0, C_j - d_j\}\}$ (cf. Fig. 1 for the common due date case). The total late work is equal to the sum of late parts of all jobs in the system

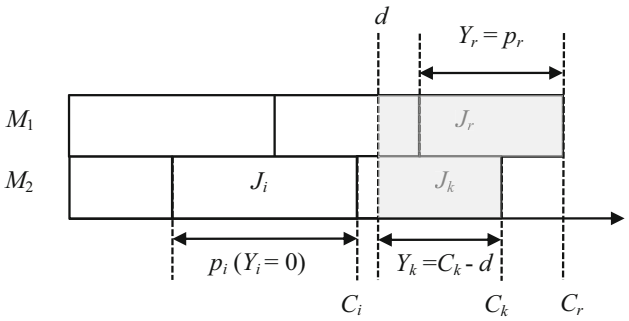

Fig. 1 Late work definition for particular jobs

i.e., $Y = \sum_{j=1}^{n} Y_j = \sum_{j=1}^{n} \min\{p_j, \max\{0, C_j - d_j\}\}$. In general, for different weights the total weighted late work criterion Y_w might be considered, i.e., $Y_w = \sum_{j=1}^{n} w_j Y_j = \sum_{j=1}^{n} w_j \min\{p_j, \max\{0, C_j - d_j\}\}$.

In the offline version, $P|d_j = d|Y$ [according to the three-field classification scheme (Graham et al. 1979)], the set of all jobs to be scheduled is known in advance. Moreover, we assume that all jobs are available at time zero (i.e., job release times are equal to zero, $r_j = 0$ for $j = 1, \ldots, n$).

In the online version, $P|d_j = d, online\ over\ list|Y$, jobs arrive in the system "over list." This means that the set of all jobs is unknown in advance, and the next job might appear in the system only after scheduling the previous one.

The late work scheduling problems model many practical situations (Sterna 2011). For example, jobs may represent customer orders in internet shops (cf. Wojciechowski and Musial 2010; Blazewicz and Musial 2011; Blazewicz et al. 2014) or tasks in production systems (cf. Sterna 2007b), which have to be executed on identical machines working in parallel, e.g., by workers having the same qualifications or on identical stages in a flexible manufacturing system, within given time, e.g., before a shipping term or within a planning horizon. The late work models the amount of work not executed on time, which determines, e.g., the loss of income caused by not executing parts of jobs on time, the fine which has to be paid to customers in case of delay, or just the amount of work which has to be scheduled in the following interval, because it was not completed in the assumed planning horizon. Jobs may also model pieces of information (Blazewicz 1984) which have to be collected by sensing devices before a given deadline. Minimizing late work corresponds to minimizing information loss, directly influencing the efficiency of control algorithms. Other applications arise in agriculture, where stretches of land have to be harvested before a given time resulting from the vegetation cycle (Potts and Van Wassenhove 1991). In such a case, late work models perished goods not harvested on time. Depending on the real world conditions, i.e., whether the situation is static (all jobs are known) or dynamic (jobs arrive one after the other), the offline or online version of the model should be applied.

Late work parameter, Y_j, was first introduced by Blazewicz (1984), who called it *information loss*, referring to a possible application of this performance measure in control systems. The phrase *late work* was proposed by Potts and Van Wassenhove (1991), who denoted this parameter as V_j, which appears in the literature alongside Y_j. The relation between late work and other performance measures (such as makespan $C_{\max}$, maximum lateness $L_{\max}$, mean (weighted) tardiness $\overline{D}/\overline{D_w}$, earliness $\overline{E}/\overline{E_w}$, and flow time $\overline{F}/\overline{F_w}$, as well as the (weighted) number of tardy jobs U/U_w) was determined by Blazewicz et al. (2000), who showed that problems with the late work criterion are at least as difficult as the analogous problems with the maximum lateness performance measure.

The majority of results for the late work criterion have been obtained for single-machine problems. For example, Potts and Van Wassenhove (1991) proposed a polynomial time algorithm with the complexity $O(nlogn)$ to solve the problem $1|pmtn|Y$ and showed that its non-preemptive case is NP-hard. Hariri et al. (1995) proposed an algorithm which solves $1|pmtn|Y_w$ with the same time complexity $O(nlogn)$. Then, a fully polynomial time approximation scheme was proposed for $1||Y$, which is an example of DP-benevolent problem defined by Woeginger (2000). For the case with job release times ($1|r_j,\ pmtn|Y$), Lin and Hsu (2005) proposed a polynomial time algorithm, also with the complexity $O(nlogn)$.

There are also a series of results concerning dedicated machines problems. Blazewicz et al. proved the binary NP-hardness of open, flow and job shop problems: $O2|d_j = d|Y_w$ (Blazewicz et al. 2004a), $F2|d_j = d|Y_w$ (Blazewicz et al. 2005a), and $J2|d_j = d, n_j \leq 2|Y_w$ (Blazewicz et al. 2007). They gave NP-hardness proofs and three dynamic programming methods with pseudo-polynomial complexity $O(nd^3)$, $O(n^2d^4)$, and $O(n^3d^{11})$, respectively. Besides this research group, Leung (2004) proved that when jobs have two distinct due dates, problems $O2|d_j \in \{d_1, d_2\}|Y$ and $F2|d_j \in \{d_1, d_2\}|Y$ are NP-hard. Using a similar approach, Lin et al. (2006) proved that $F2|d_j = d|Y$ is NP-hard. In addition to these theoretical results, computational experiments with metaheuristic approaches for shop systems were reported in the literature (cf., e.g., Blazewicz et al. 2005b, c; Pesch and Sterna 2009).

Although the late work criterion was originally proposed for parallel machines, only a few results have been obtained for this machine environment so far. To the best of our knowledge, only three papers focus directly on this subject. Blazewicz (1984) and Blazewicz and Finke (1987) investigated problems $P||Y$, $P|r_j,\ pmtn|Y_w$ and $Q|r_j,\ pmtn|Y_w$. Then, Leung (2004) gave some results for unweighted cases $P|r_j,\ pmtn|Y$ and $Q|r_j,\ pmtn|Y$. The non-preemptive problem mentioned above is NP-hard, while the preemptive ones are polynomially solvable.

3 Offline problem $P|d_j = d|Y$

The complexity status of the offline scheduling problem $P2|d_j = d|Y$ has not been determined so far, despite the fact that it might be quite easily predicted. Based on the literature, we know that the problem with an arbitrary number of machines and arbitrary due dates, $P||Y$, is NP-hard (Blazewicz 1984). Actually, already the single-machine problem, $1||Y$, is binary NP-hard (Potts and Van Wassenhove 1991). In consequence, the two-machine problem with arbitrary due dates, $P2||Y$, has to be at least binary NP-hard. By contrast, the single-machine problem with a common due date, $1|d_j = d|Y$, is polynomially solvable, since any schedule without idle time is optimal (Potts and Van Wassenhove 1991). There is no known result concerning the complexity of the two-machine non-preemptive problem with a common due date. On the other hand, it is known that late work scheduling problems are at least as difficult as the problems of minimizing the makespan (Blazewicz et al. 2000). We know that the makespan minimization problem on two identical parallel machines, $P2||C_{\max}$, is binary NP-hard (Garey and Johnson 1979). But this result does not determine the complexity of the problem $P2|d_j = d|Y$ (but $P2||Y$). Nevertheless, there is a very close relation between $P2||C_{\max}$ and $P2|d_j = d|Y$.

Theorem 1 *In the offline case, any optimal solution for $P2||C_{max}$ is optimal for $P2|d_j = d|Y$.*

Proof Consider the schedule which is optimal for problem $P2||C_{\max}$. Denote the makespan on M_i as C^i ($i = 1$ and 2). Then the optimal schedule length is equal to $C^*_{\max} = \max\{C^1, C^2\}$. Let us assume without loss of generality that $C^1 = C^*_{\max}$; therefore, $C^2 \leq C^1$.

Case 1 If $d \geq C^1$, then all jobs in the schedule are early, the total late work equals zero ($Y = 0$), and the schedule is optimal for $P2|d_j = d|Y$.
Case 2 If $d < C^1$ and $d \geq C^2$, then $Y = C^1 - d = C^*_{\max} - d$, and it cannot be smaller, since $C^*_{\max}$ is minimal; therefore, the schedule is optimal for $P2|d_j = d|Y$.
Case 3 If $d < C^2$, then there are late jobs on both machines and the total late work equals $Y = (C^1 - d) + (C^2 - d) = (C^*_{\max} - d) + (C^2 - d) > 0$. Since jobs are executed without idle time and machines are busy in the whole interval $[0,\ d]$, the amount of early work cannot be increased; thus, the late work cannot be decreased, so the schedule is optimal for $P2|d_j = d|Y$. □

Theorem 1 shows that using a method of solving the problem with the makespan criterion, which is NP-hard, we can also solve the problem with the late work criterion and a common due date. But it does not mean that the solution for the late work criterion cannot be found in polynomial time.

The NP-hardness of problem $P2|d_j = d|Y$ results from its similarity to the Partition Problem, which is NP-complete (Garey and Johnson 1979).

Theorem 2 *The problem $P2|d_j = d|Y$ is NP-hard.*

Proof The decision counterpart of problem $P2|d_j = d|Y$ obviously belongs to NP, since its solution can be verified in polynomial time.

Consider the NP-complete *Partition Problem*, formulated as follows: There is given a set $A = \{a_1, \ldots, a_n\}$ of n elements of size s_j. The question is does there exist a subset $A' \subseteq A$, such that $\sum_{a_j \in A'} s_j = \sum_{a_j \in A \setminus A'} s_j$?

Consider the scheduling problem with two identical parallel machines and a common due date, $P2|d_j = d|Y$, where n jobs J_j $(1 \le j \le n)$ from set J correspond to n elements a_j from A, i.e., $p_j = s_j$, and $d = \frac{1}{2}\sum_{a_j \in A} s_j = \frac{1}{2}\sum_{j=1}^{n} p_j$. The question is does there exist a solution for $P2|d_j = d|Y$ with the total late work equal to zero $(Y = 0)$?

Both problems are equivalent. If the Partition Problem has a solution, then we can execute jobs corresponding to $A \setminus A'$ on M_1 and to A' on M_2. The schedule length on both machines is equal to $\frac{1}{2}\sum_{a_j \in A} s_j = \frac{1}{2}\sum_{j=1}^{n} p_j$, and all jobs are early, leading to zero total late work.

On the other hand, if there is a schedule with zero late work, then all jobs finish not later than at common due date d. Since $d = \frac{1}{2}\sum_{j=1}^{n} p_j$, the jobs have to be executed on both machines without idle times exactly for d time units. The division of jobs between machines determines the solution of the Partition Problem. □

Moreover, problem $P2|d_j = d|Y$ is binary NP-hard, because it can be solved in pseudo-polynomial time by a simple dynamic programming method. Assuming that $f(j, A, B)$ denotes the total late work for j jobs scheduled on the machines, where fully early jobs are executed for at most A and B units on M_1 and M_2, respectively, the optimal total late work is equal to $f(n, d, d)$. The criterion value can be determined by the recurrence function with zero initial conditions:

$$f(j, A, B) = \min\Big\{f\big(j-1, \max\{0, A - p_j\}, B\big) + \max\{0, p_j - A\}, f\big(j-1, A, \max\{0, B - p_j\}\big) + \max\{0, p_j - B\}\Big\}.$$

The first formula represents schedules with job J_j assigned to machine M_1, while the second one represents schedules with J_j executed on M_2. Since jobs are scheduled without idle time on both machines and their sequence is not important from the criterion value point of view, then the solution process is reduced to assigning jobs to machines or—in other words—to packing jobs before the due date. Because the recurrence function has to be determined for n jobs and $0 \le A \le d$, $0 \le B \le d$, the method runs in pseudo-polynomial time $O(nd^2)$.

Taking into account the fact that the two-machine case, $P2|d_j = d|Y$, is NP-hard, its generalization for an arbitrary number of machines $P|d_j = d|Y$ is also NP-hard (Garey and Johnson 1979).

Actually, using analogous reasoning as in Theorem 2, it is easy to show that problem $P|d_j = d|Y$ is unary NP-hard due to the transformation from the 3-Partition Problem. The general idea of the proof is as follows.

In the unary NP-complete 3-*Partition Problem* (Garey and Johnson 1979), there is given a set $A = \{a_1, \ldots, a_{3n}\}$ of $3n$ elements of size s_j, such that $\sum_{a_j \in A} s_j = nB$, where $\frac{1}{4}B < s_j < \frac{1}{2}B$ for each $j = 1, \ldots, 3n$. The question is does there exist a partition of A into $A_1, \ldots, A_n$, such that $\sum_{s_j \in A_i} s_j = B$ for each $i = 1, \ldots, n$? Constructing a schedule with $3n$ jobs, corresponding to elements a_j, on n machines with zero total late work with regard to common due date $d = B$, is equivalent to solving the 3-Partition Problem.

Theorems 1 and 2 show the similarity of problems $P2|d_j = d|Y$ and $P2||C_{\max}$. We know that solutions minimizing makespan are optimal from the total late work point of view, and no other schedule minimizing the total late work can be constructed in polynomial time. Moreover, the dynamic programming method proposed above shows the similarity of the considered scheduling problem to the *Knapsack Problem* or the *Bin Packing Problem*, because we try to pack as many time units of jobs before a common due date as possible.

Although problem $P2|d_j = d|Y$ is closely related to these classical combinatorial problems, determining its computational complexity, and consequently the complexity of $P|d_j = d|Y$, was necessary to start research on online versions of these cases, because the efficiency of online algorithms is evaluated based on the comparison of online and offline solutions.

4 Online problem $P|d_j = d$, *online over list*$|Y$

In the literature (cf., e.g., Tan and Zhang 2013), two basic models of online scheduling are discussed: 1) *online "over list"* model and 2) *online "over time"* one. In online "over list" scheduling, it is assumed that jobs come into the system one after another, i.e., the information on the next job becomes available—without any delay—after the previous one has been processed . In online "over time" scheduling, each job has its release time, and the information on this job becomes known only after this time.

Within the reported research, we focus on the first branch of online scheduling. All the problem parameters defined for the offline case in Sect. 3 apply to the online case, too.

Particularly, each job J_j has processing time p_j and all jobs should be preferably executed before common due date d. Our goal is to schedule all jobs on m identical machines, so that the total late work Y is minimized.

Since the set of jobs is unknown in advance, a schedule has to be constructed by an online algorithm. Obviously, due to incomplete knowledge about the whole set of jobs, online schedule might be worse than an optimal offline schedule, determined under perfect knowledge about all problem parameters.

To estimate the quality of online scheduling algorithms, we often use *competitive ratio*—a classic measure which shows how close an online solution is to an offline optimal solution (cf., e.g., Borodin and El-Yaniv 1998; Fiat and Woeginger 1998). For example, in a scheduling problem with an objective function which is minimized, for an input π and an online algorithm A, where $C^A_{\max}(\pi)$ denotes the criterion value produced by A, and $C^*_{\max}(\pi)$ denotes the optimal solution value in an offline model, the *competitive ratio* of A is the infimum r such that for any input, $C^A_{\max}(\pi) \leq r \cdot C^*_{\max}(\pi)$. We call A an $r-competitive$ algorithm.

Moreover, an online problem has a *lower bound* ρ, if no online scheduling algorithm has a competitive ratio strictly smaller than ρ.

An online scheduling algorithm is called *optimal*, if its competitive ratio is equal to this lower bound.

However, as we see from the following lemma, we cannot use the competitive ratio to estimate the quality of online scheduling algorithms minimizing late work directly.

Lemma 3 *There is no constant lower bound of competitive ratio for* $P|d_j = d, online\ over\ list|Y$.

Proof Denote the total late work produced by an online algorithm A as Y^A and the optimal total late work for an offline solution as Y^*. Let $m = 2$ and $d = 2-\epsilon$, where $0 < \epsilon < 1$. Assume that the first two jobs which arrive in the system have unit processing times $p_1 = p_2 = 1$.

Case 1 Assume that these two jobs have been assigned by any online algorithm to the same machine. Then the job sequence ends and no more jobs arrive in the system. Hence, we get $Y^A = 2-d = \epsilon$. But in the optimal offline solution, we have $Y^* = 0$, by assigning the jobs to different machines.

Case 2 Now assume that this online algorithm has assigned these two jobs to different machines; then the third job with $p_3 = 2$ comes. We must assign it to one of the machines, and we have $Y^A = 3-d = 1+\epsilon$. But in the optimal offline solution, we have $Y^* = (2-d)+(2-d) = 2\epsilon$, by assigning the first two jobs to one machine and the third one to the other. Then $\frac{Y^A}{Y^*} = \frac{1+\epsilon}{2\epsilon} = \frac{1}{2\epsilon} + \frac{1}{2} \to \infty$, when $\epsilon \to 0$. □

From Lemma 3 we see that the competitive ratio calculated based on the total late work is useless for investigating online problems, since it might be not well defined (as in Case 1) or it might be infinite (as in Case 2), and another approach should be applied. Actually, to determine the competitive ratio for online algorithms for $P|d_j = d, online\ over\ list|Y$, we can use the concept of $early\ work\ (X)$, which is of course complementary to late work (Blazewicz et al. 2005a). Early work denotes the part of job J_j executed before (instead after) the due date (X_j) and any algorithm solving the problem should maximize the total early work $(X = \sum_{j=1}^{n} X_j)$. For the offline problem both concepts are fully equivalent, but for the online case only early work allows us to investigate the competitive ratio. As it was shown, the total late work could be zero in an optimal offline solution, and, in this case, we could not use the competitive ratio to estimate the quality of an online solution (we cannot use zero as denominator). On the contrary, early work is always positive (except for the trivial cases with $d = 0$ or $n = 0$).

Hence, we will denote with X^A the total early work of a solution constructed by the online algorithm A and with X^* the criterion value of an optimal offline solution. Then the competitive ratio for the early work scheduling problem can be defined as infimum r such that for any input $\frac{X^*}{X^A} \leq r$.

4.1 Algorithm for $P|d_j = d, online\ over\ list|Y$

To solve the considered problem, we propose an online algorithm called EFF_m (*Extended First Fit for m machines*), with the competitive ratio $\frac{\sqrt{2m^2-2m+1}-1}{m-1}$. We use the following notation:

- L^i_j: the load on machine M_i $(i = 1, \ldots, m)$ after job J_j $(j = 1, \ldots, n)$ has been assigned (i.e., current makespan on M_i),
- Sum: the total size of all jobs, $Sum = \sum_{j=1}^{n} p_j$,
- X^{EFF_m}: the total early work of online solution constructed by EFF_m,
- X^*: the optimal early work of offline solution (note that $X^* \leq \min\{Sum, md\}$ from the definition of early work),
- $r_m = \frac{\sqrt{2m^2-2m+1}-1}{m-1}$: the desired competitive ratio $(m \geq 2)$.

The online algorithm EFF_m assigns a new job to the first suitable machine or to the machine with the minimum load, if there is no suitable one. The machine is suitable, if after assigning a new job its load will not exceed the assumed ratio, i.e., $r_m d$.

Algorithm EFF_m

1. Set $t = 1$, $L^i_0 = 0$ for $i = 1, \ldots, m$.
2. When job J_t comes, assign it to the first machine which fits it, without violating the ratio $(First\ Fit)$, i.e., for $(i = 1; i \leq m; i++)$

if $(L^i_{t-1} + p_t \le r_m d)\{$
assign J_t to M_i (i.e., $L^i_t = L^i_{t-1} + p_t$)
break;
$\}$.

3. If $(i > m)$, then assign J_t to the machine with the minimum load.
4. If there is another job in the input sequence, set $t = t + 1$ and go to *Step* 2, else stop.

Theorem 4 *The competitive ratio of Algorithm EFF_m is $r_m = \frac{\sqrt{2m^2-2m+1}-1}{m-1}$ $(m \ge 2)$.*

Proof For the sake of simplicity, we assume that the machines are numbered in the non-increasing order of their loads when Algorithm EFF_m stops.
Consider the time after the last job J_n has been assigned to a machine.
Case 1 If $\max_{i=1,\ldots,m}\{L^i_n\} \le d$, we have $X^{EFF_m} = Sum$, and the solution is optimal.
Case 2 If $\min_{i=1,\ldots,m}\{L^i_n\} \ge d$, we have $X^{EFF_m} = md$, and the solution is also optimal.
So we focus on *Case 3* with

$$\min_{i=1,\ldots,m}\{L^i_n\} < d < \max_{i=1,\ldots,m}\{L^i_n\}$$

Subcase 3.1:

$$\min_{i=1,\ldots,m}\{L^i_n\} < d < \max_{i=1,\ldots,m}\{L^i_n\} \le r_m d$$

Assume there are k machines with late work (i.e., their load exceeds d). Then we have $\frac{X^*}{X^{EFF_m}} \le \frac{Sum}{X^{EFF_m}} = \frac{\sum_{j=1}^{k} L^j_n + \sum_{j=k+1}^{m} L^j_n}{kd + \sum_{j=k+1}^{m} L^j_n}$. Note $\frac{a+c}{b+c} \le \frac{a}{b}$, when $a \ge b > 0$ and $c \ge 0$, so we have $\frac{X^*}{X^{EFF_m}} \le \frac{\sum_{j=1}^{k} L^j_n}{kd} \le \frac{kr_m d}{kd} = r_m$.
Subcase 3.2:

$$\min_{i=1,\ldots,m}\{L^i_n\} < d < r_m d < \max_{i=1,\ldots,m}\{L^i_n\}$$

Let J_L be the last job on the machine with the maximum load. Then this machine had the minimum load (let denote this load with A) before assigning J_L (according to *Step* 3 of Algorithm EFF_m). Let $B \ge A$ be the second smallest machine load before assigning job J_L.

If $A > 0$, then A and B contain processing times of some jobs, and we have $A + B > r_m d$; else these jobs would have been assigned to the same machine according to *First Fit* rule applied within EFF_m. Since $B \ge A$, we have $B > \frac{1}{2} r_m d$. Then we have $X^{EFF_m} \ge (m-1)B + d$, since there is at least one machine with late work and others with load greater than B. Then $\frac{X^*}{X^{EFF_m}} \le \frac{md}{(m-1)B+d} < \frac{md}{\frac{(m-1)r_m d}{2}+d} = \frac{2m}{(m-1)r_m+2}$.

According to its definition, $r_m = \frac{\sqrt{2m^2-2m+1}-1}{m-1}$, so we have $\frac{X^*}{X^{EFF_m}} < \frac{2m}{\sqrt{2m^2-2m+1}+1} = \frac{\sqrt{2m^2-2m+1}-1}{m-1} = r_m$.
Otherwise $A = 0$, which means that there is a "big" job with processing time $p > r_m d$. Exclude temporarily this job and the machine on which it has been scheduled from the analysis. We get a new job sequence J' and a new schedule for it. Let X' be the total early work of this new schedule constructed by EFF_m and let X'^* be the optimal offline early work for this instance. Using the reasoning presented before, we can show that $\frac{X'^*}{X'} \le r_m$. Then we have $\frac{X^*}{X^{EFF_m}} = \frac{X'^*+d}{X'+d} \le r_m$, since $r_m \ge 1$. □

Lemma 5 *The competitive ratio of Algorithm EFF_m, i.e., $r_m = \frac{\sqrt{2m^2-2m+1}-1}{m-1} = \frac{\sqrt{(m-1)^2+m^2}-1}{m-1}$, is an increasing function of m.*

Proof To prove this lemma, it is sufficient to prove that $r_{m+1} > r_m$, i.e.,

$$\begin{aligned}
& r_{m+1} > r_m \\
\Leftrightarrow\ & \frac{\sqrt{m^2+(m+1)^2}-1}{m} > \frac{\sqrt{(m-1)^2+m^2}-1}{m-1} \\
\Leftrightarrow\ & (m-1)(\sqrt{m^2+(m+1)^2}-1) \\
& > m(\sqrt{(m-1)^2+m^2}-1) \\
\Leftrightarrow\ & (m-1)(\sqrt{m^2+(m+1)^2}) \\
& > m\sqrt{(m-1)^2+m^2}-1 \\
\Leftrightarrow\ & (m-1)^2(m^2+(m+1)^2) \\
& > m^2((m-1)^2+m^2) - 2m\sqrt{(m-1)^2+m^2}+1 \\
\Leftrightarrow\ & (m-1)^2(m+1)^2 \\
& > m^4 - 2m\sqrt{(m-1)^2+m^2}+1 \\
\Leftrightarrow\ & m^4 - 2m^2 + 1 > m^4 - 2m\sqrt{(m-1)^2+m^2}+1 \\
\Leftrightarrow\ & \sqrt{(m-1)^2+m^2} > m \\
\Leftrightarrow\ & (m-1)^2 + m^2 > m^2 \\
\Leftrightarrow\ & (m-1)^2 > 0.
\end{aligned}$$

Thus, this lemma holds for $m \ge 2$. □

Taking into account that for $m \to \infty$, the competitive ratio of the proposed method converges to $r_\infty = \sqrt{2} \approx 1.414$. Hence, we can state that the competitive ratio of Algorithm EFF_m is bounded by constant $\sqrt{2}$.

4.2 Lower bound of $P2|d_j = d, online\ over\ list|Y$

Now, we show that the lower bound of problem $P2|d_j = d, online\ over\ list|Y$ is equal to $\sqrt{5} - 1 \approx 1.236$.

Theorem 6 *For $P2|d_j = d, online\ over\ list|Y$, no online algorithm has its competitive ratio strictly less than $\sqrt{5} - 1$.*

Proof Let $d = \frac{1+\sqrt{5}}{2}$ and assume that the first two jobs appearing in the system have unit processing times. There are only two possible ways of scheduling them.

Case 1 If these jobs have been assigned to the same machine, then the input job sequence ends. We have $X^A = d$, while $X^* = 2$, and $\frac{X^*}{X^A} = \frac{2}{d} = \sqrt{5} - 1$.
Case 2 If these two jobs have been assigned to different machines, then the last job with processing time equal to 2 comes into the system. We have $X^A = 1 + d$ and $X^* = 2d$, then $\frac{X^*}{X^A} = \frac{2d}{1+d} = \frac{1+\sqrt{5}}{1+\frac{1+\sqrt{5}}{2}} = \sqrt{5} - 1$. □

Taking into account the fact that for two machines Algorithm EFF_m has the competitive ratio r_2 which is equal to the lower bound presented above, EFF_2 is an optimal online algorithm.

5 Conclusions

The scheduling problems with the late work criterion have been investigated for nearly 30 years, but most of the literature focused on single-machine and shop environments. In this paper we returned to the parallel machines environment, for which this performance measure was originally proposed. We investigated, for the first time, the online model, initiating studies on online versions of late work minimization problems.

Namely, for the offline case, we established the computational complexity of the problem with a common due date and an arbitrary number of machines, showing its unary NP-hardness, and we proved binary NP-hardness of the two-machine case. The research on the offline problems might be continued, as for others intractable problems, by exploring the structure of their optimal offline solutions (cf., e.g., Sterna 2007a) or by designing heuristic/metaheuristic approaches (cf., e.g., Blazewicz et al. 2004b, 2008).

For the online case, the constant competitive ratio algorithm was given for an arbitrary number of machines, which appears to be optimal for two identical machines. Since the online environment has not been taken into account in the context of late work minimization so far, the scope for future research is overwhelming. For the parallel machines case with a common due date, one can take into account—for example—semi-online problems, allowing some jobs rearrangements (cf., e.g., Tan and Yu 2008; Chen et al. 2011) or assuming existence of buffers (cf., e.g., Englert et al. 2008; Lan et al. 2012).

The natural step would be also investigating other problems, whose offline versions' complexity status has been already determined, in the online mode.

Acknowledgements This research was partially supported by FNR (Luxembourg) and NCBiR (Poland) through IShOP Project INTER/POLLUX/13/6466384, NSFC (61300016, 11101065), and China Scholarship Council (CSC). We express our gratitude to the anonymous Referees, whose valuable suggestions and constructive comments allowed us to improve this paper.

References

Blazewicz, J. (1984). Scheduling preemptible tasks on parallel processors with information loss. *Technique et Science Informatiques*, *3*(6), 415–420.
Blazewicz, J., Bouvry, P., Kovalyov, M.Y., & Musial, J. (2014). Internet shopping with price sensitive discounts. *4OR-A Quarterly Journal of Operations Research, 12*(1), 35–48.
Blazewicz, J., & Finke, G. (1987). Minimizing mean weighted execution time loss on identical and uniform processors. *Information Processing Letters*, *24*(4), 259–263.
Blazewicz, J., & Musial, J. (2011). E-commerce evaluation – multi-item internet shopping. Optimization and heuristic algorithms. In B. Hu, K. Morasch, S. Pickl & M. Siegle (Eds.), *Operations research proceedings 2010* (pp. 149–154), Berlin: Springer.
Blazewicz, J., Pesch, E., Sterna, M., & Werner, F. (2000). Total late work criteria for shop scheduling problems. In K. Inderfurth, G. Schwoediauer, W. Domschke, F. Juhnke, P. Kleinschmidt, & G. Waescher (Eds.), *Operations research proceedings 1999* (pp. 354–359). Berlin: Springer.
Blazewicz, J., Pesch, E., Sterna, M., & Werner, F. (2004a). Open shop scheduling problems with late work criteria. *Discrete Applied Mathematics*, *134*(1), 1–24.
Blazewicz, J., Pesch, E., Sterna, M., & Werner, F. (2004b). Flow shop scheduling with late work criterion - choosing the best solution strategy. *Lecture Notes in Computer Science*, *3285*, 68–75.
Blazewicz, J., Pesch, E., Sterna, M., & Werner, F. (2005a). The two-machine flow-shop problem with weighted late work criterion and common due date. *European Journal of Operational Research*, *165*(2), 408–415.
Blazewicz, J., Pesch, E., Sterna, M., & Werner, F. (2005b). Metaheuristics for late work minimization in two-machine flow shop with common due date. *Lecture Notes in Artificial Intelligence*, *3698*, 222–234.
Blazewicz, J., Pesch, E., Sterna, M., & Werner, F. (2005c). A comparison of solution procedures for two-machine flow shop scheduling with late work criterion. *Computers and Industrial Engineering*, *49*(4), 611–624.
Blazewicz, J., Pesch, E., Sterna, M., & Werner, F. (2007). A note on the two machine job shop with the weighted late work criterion. *Journal of Scheduling*, *10*(2), 87–95.
Blazewicz, J., Pesch, E., Sterna, M., & Werner, F. (2008). Metaheuristic approaches for the two-machine flow-shop problem with weighted late work criterion and common due date. *Computers and Operations Research*, *35*(2), 574–599.
Borodin, A., & El-Yaniv, R. (1998). *Online computation and competitive analysis*. Cambridge: Cambridge University Press.
Chen, X., Lan, Y., Benko, A., Dosa, G., & Han, X. (2011). Optimal algorithms for online scheduling with bounded rearrangement at the end. *Theoretical Computer Science*, *412*(45), 6269–6278.
Emmons, H. (1969). One-machine sequencing to minimize certain functions of job tardiness. *Operations Research*, *17*(4), 701–715.
Englert, M., Ozmen, D., & Westermann, M. (2008). The power of reordering for online minimum makespan scheduling. In *Proceed-*

ings of 48th IEEE symposium foundations of computer science (FOCS) (pp. 603–612).

Fiat, A., & Woeginger, G. (Eds.) (1998). Online algorithms: The state of the art. *Lecture Notes in Computer Science, 1442*.

Garey, M., & Johnson, D. (1979). *Computers and Intractability: A guide to the theory of NP-Completeness*. New York: WH Freeman and Company.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, *5*, 287–326.

Hariri, A. M. A., Potts, C. N., & Van Wassenhove, L. N. (1995). Single machine scheduling to minimize total weighted late work. *ORSA Journal on Computing*, *7*(2), 232–242.

Hochbaum, D. S., & Shamir, R. (1990). Minimizing the number of tardy job units under release time constraints. *Discrete Applied Mathematics*, *28*(1), 45–57.

Kethley, R. B., & Alidaee, B. (2002). Single machine scheduling to minimize total weighted late work: A comparison of scheduling rules and search algorithms. *Computers and Industrial Engineering*, *43*(3), 509–528.

Kovalyov, M. Y., Potts, C. N., & Van Wassenhove, L. N. (1994). A fully polynomial approximation scheme for scheduling a single machine to minimize total weighted late work. *Mathematics of Operations Research*, *19*(1), 86–93.

Lan, Y., Chen, X., Ding, N., Dosa, G., & Han, X. (2012). Online minimum makespan scheduling with a buffer. In J. Snoeyink, P. Lu, K. Su, & L. Wang (Eds.), *Proceeding of FAW-AAIM* (pp. 161–171). Beijing: Springer.

Leung, J. Y. T. (2004). Minimizing total weighted error for imprecise computation tasks and related problems. In J. Y. T. Leung (Ed.), *Handbook of scheduling: algorithms, models, and performance analysis* (pp. 34.1–34.16). Boca Raton: CRC Press.

Lin, BMT., Hsu, SW. (2005). Minimizing total late work on a single machine with release and due dates. In *SIAM conference on computational science and engineering*, Orlando.

Lin, B. M. T., Lin, F. C., & Lee, R. C. T. (2006). Two-machine flow-shop scheduling to minimize total late work. *Engineering Optimization*, *38*(4), 501–509.

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

McMahon, G., & Florian, M. (1975). On scheduling with ready times and due dates to minimize maximum lateness. *Operations Research*, *23*(3), 475–482.

Pesch, E., & Sterna, M. (2009). Late work minimization in flow shop by a genetic algorithm. *Computers and Industrial Engineering*, *57*(4), 1202–1209.

Potts, C. N., & Van Wassenhove, L. N. (1991). Single machine scheduling to minimize total late work. *Operations Research*, *40*(3), 586–595.

Ren, J., Zhang, Y., & Sun, G. (2009). The NP-hardness of minimizing the total late work on an unbounded batch machine. *Asia-Pacific Journal of Operational Research*, *26*(3), 351–363.

Sterna, M. (2007). Dominance relations for two-machine flow-shop problem with late work criterion. *Bulletin of the Polish Academy of Sciences*, *55*(1), 59–69.

Sterna, M. (2007). Late work minimization in a small flexible manufacturing system. *Computers and Industrial Engineering*, *52*(2), 210–228.

Sterna, M. (2011). A survey of scheduling problems with late work criteria. *Omega*, *39*(2), 120–129.

Tan, Z., & Yu, S. (2008). Online scheduling with reassignment. *Operations Research Letters*, *36*(2), 250–254.

Tan, Z., & Zhang, A. (2013). Online and semi-online scheduling. In P. M. Pardalos, et al. (Eds.), *Handbook of Combinatorial Optimization* (pp. 2191–2252). New York: Springer.

Woeginger, G. J. (2000). When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing*, *12*(1), 57–74.

Wojciechowski, A., & Musial, J. (2010). Towards optimal multi-item shopping basket management: Heuristic approach. *Lecture Notes in Computer Science, 6428*, 349–357.

Zhang, Y., & Wang, L. (2005). The NP-completeness of a new batch scheduling problem. *Journal of Systems Science and Mathematical Sciences*, *25*(1), 13–17.